Uso de Escenarios

en la Derivación de Software

Tesis Doctoral

Graciela Dora Susana Hadad

Director de tesis: *Prof. Dr. Julio Cesar Sampaio do Prado Leite*
Pontificia Universidade Catolica do Rio de Janeiro

Asesor Científico: *Prof. Ing. Jorge Horacio Doorn*
Universidad Nacional de La Plata

Facultad de Ciencias Exactas
Universidad Nacional de La Plata – Argentina

Resumen

Esta tesis presenta una estrategia en la Ingeniería de Requisitos, denominada SDRES, que intenta abordar temas poco tratados en la práctica real, tales como los cambios constantes en los requisitos, defectos del software originados en los requisitos, el contexto organizacional que rodea al sistema de software y la consideración de requisitos de calidad.

Esta estrategia está dirigida por modelos (Léxico Extendido del Lenguaje, Escenarios y Documento de Requisitos) y orientada al cliente, por ello utiliza sus modelos escritos en lenguaje natural como medio de comunicación y elicitación. SDRES tiene en cuenta la calidad de los modelos que produce mediante procesos de verificación y validación. Para cada actividad de la estrategia se presenta un conjunto de heurísticas y recomendaciones. Se encara el tema de evolución y versionado de los modelos, así como distintas modalidades de utilizar la estrategia según la complejidad del problema, el conocimiento sobre el mismo y otras características.

Abstract

The present thesis shows a Requirements Engineering strategy, called SDRES, which proposes to face topics rarely treated in real practice, such as continuous changes in requirements, software defects brought in requirements, the organisational context surrounding the software system and the quality treatment of requirements.

This strategy is driven by models (Language Extended Lexicon, Scenarios and Software Requirements Specification) and oriented to the client. Therefore it uses models written in natural language as means of communication and elicitation. SDRES keeps in mind the quality of the produced models by means of verification and validation processes. For each activity of the strategy a set of heuristic and recommendations is presented. The evolution topic and model versioning is treated, as well as different modalities to use the strategy according to the complexity of the problem, the knowledge on the problem and other characteristics.

«Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away. »

Antoine de Saint Exupery

Índice

Lista de Figuras

Lista de Tablas

Lista de Acrónimos utilizados

ACM	Association for Computer Machinery
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CORB	Client-Oriented Requirements Baseline
COTS	Commercial Off The Shelf
CREWS	Cooperative Requirements Engineering With Scenarios
DEO	Discrepancias, Errores y Omisiones
DER	Diagrama de Entidad-Relación
DFD	Diagrama de Flujo de Datos
EA	Escenarios Actuales
EF	Escenarios Futuros
EG	Escenarios Generales
EI	Escenarios Integradores
ESPRIT	European Strategic Program on Information Technology
IEC	International Electrotechnical Commission
IEEE	Institute for Electrical and Electronics Engineers
IID	Iterative and Incremental Development
IR	Ingeniería de Requisitos
ISO	International Standardisation Organisation
JAD	Joint Application Design
LEL	Language Extended Lexicon / Léxico Extendido del Lenguaje
LN	Lenguaje Natural
MD	Modelo de Diseño
NATO	North Atlantic Treaty Organisation
OMG	Object Management Group
OMT	Object Modeling Technique
RAD	Rapid Application Development
RE	Requirements Engineering
RF	Requisitos Funcionales
RNF	Requisitos No Funcionales
RUP	Rational Unified Process
SADT	Structured Analysis and Design Technique
SDRES	Scenario Driven Requirements Engineering Strategy
SEI	Software Engineering Institute
SPICE	Software Process Improvement and Capability Evaluation
SRS	Software Requirements Specification
TSE	Transactions on Software Engineering
UB	Universidad de Belgrano - Argentina
UdeD	Universo de Discurso
UML	Unified Modelling Language
UNCPBA	Universidad Nacional del Centro de la Provincia de Buenos Aires

UNLaM	Universidad Nacional de La Matanza
UTN	Universidad Tecnológica Nacional
V&V	Verificación y Validación
XP	eXtreme Programming / Programación eXtrema

Capítulo 1
Introducción

«The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.»

F. Brooks, "No Silver Bullet", IEEE Computer, 1987

1.1. Motivación

1.1.1. La importancia de los requisitos

La clave del éxito o fracaso de un proyecto de software depende fuertemente de **resolver el problema correcto** [Rumbaugh 94], es decir, encontrar el conjunto de soluciones adecuadas al problema en el universo de discurso (UdeD)[1]. Para lo cual, se debe primero adquirir conocimiento sobre el UdeD, capturando las demandas, necesidades y problemas presentes en él y, luego, determinar que los requisitos especificados para el sistema de software sean los correctos.

Los errores en los requisitos incrementan altamente los costos de desarrollo y mantenimiento de los sistemas de software, dado que la corrección de los mismos puede involucrar desde simples adaptaciones al código, el rediseño de algún componente o hasta reformular total o parcialmente la solución propuesta. Es sabido que los costos por corrección de errores en los requisitos se incrementan en cada fase de desarrollo del software. La incidencia del costo de corrección de requisitos erróneos ha sido ampliamente estudiada por autores como Michael Fagan [Fagan 74], Barry Boehm [Boehm 76], Bell y Thayer [Bell 76], Edmund Daly [Daly 77], Victor Basili [Basili 81] y Alan Davis [Davis 93] entre otros. Barry Boehm [Boehm 81] estudió varios proyectos de desarrollo de software de gran envergadura para determinar el costo de corrección de errores en requisitos (ver Tabla 1-1), errores que eran descubiertos tardíamente en el proceso de desarrollo.

Como se puede observar en la Tabla 1-1, en el caso de un requisito erróneo que permanece sin ser detectado ni corregido hasta la etapa de operación, su reparación puede costar hasta 100 veces más que si fuera

[1] Universo de discurso [Leite 94]: todo el contexto en el cual el software se desarrolla e incluye todas las fuentes de información y toda la gente relacionada con el software. Es la realidad acotada por el conjunto de objetivos establecidos por quienes demandan una solución de software. Las personas involucradas en el proceso de desarrollo son principalmente: usuarios, clientes, ingenieros de software, expertos del dominio, las cuales son denominadas en la literatura con la palabra "stakeholders".

detectado y corregido en la fase de establecimiento de los requisitos.

Fase donde se detectó el error	Costo promedio
Definición de Requisitos	1
Diseño	3-6
Codificación	10
Pruebas de desarrollo	15-40
Prueba de aceptación	30-70
Operación	40-100

Tabla 1-1. Costo relativo de corrección de errores [Boehm 81]

Se puntualizan a continuación aquellos aspectos a considerar en los errores en los requisitos [Davis 93]:

i) *Cuanto más tarde en el ciclo de vida se detecta un error, más costoso es repararlo*.

> ✓ El crecimiento de los costos de reparación es producto de la catarata de errores que se producen: errores originados en una etapa se arrastran en fases sucesivas más nuevos errores que se originan en cada una de ellas (ver Figura 1-1).
>
> [Mizuno 83]

Se debe acotar que el costo de reparación tiene ligado naturalmente un costo de desarrollo previo de los artefactos que se desecharán o rehacerán en la cadena de errores.

ii) *Muchos errores permanecen latentes y no son detectados hasta bastante después de la etapa en que se cometieron*.

> ✓ El 54% de los errores se detectan después de la fase de codificación y prueba.
> ✓ Estos errores provienen en un 45% de la fase de requisitos y en un 9% de la fase de codificación.
>
> [Boehm 75]

iii) *Se están cometiendo demasiados errores*.

> ✓ El 56% de los errores tienen su origen en la etapa de requisitos.
>
> Informado por Tom DeMarco, citado en [Tavolato 84]

iv) *Los errores de requisitos responden a una clasificación típica.*

- ❖ hechos incorrectos 49%
- ❖ omisiones 31%
- ❖ inconsistencias 13%
- ❖ ambigüedades 5%
- ❖ otros (requisitos mal ubicados) 2%

[Basili 81]

v) *Los errores en los requisitos pueden detectarse*.

- ✓ La detección puede realizarse a través de distintas técnicas: ad hoc, inspecciones, pruebas unitarias, pruebas de integración, evaluación (pruebas de aceptación), otros.
- ✓ Las inspecciones detectan el 65% de errores, las pruebas unitarias: 10%, pruebas de integración: 6%, evaluación: 10%, otros: 10%.

[Davis 93]

Los puntos recién mencionados se pueden resumir en las siguientes conclusiones [Davis 93]:

- ⇨ Se cometen muchos errores de requisitos.
- ⇨ Muchos de esos errores se detectan tardíamente.
- ⇨ Sin embargo, muchos de esos errores pueden detectarse tempranamente.
- ⇨ No detectar los errores contribuye al incremento de costos en el software.

Diversas estadísticas realizadas a lo largo de las últimas décadas referidas a fracasos en proyectos de software [Alford 79] [Boehm 81] [GAO 92] [Faulk 92] [Lutz 93] [CHAOS 95] concluyen que los requisitos son la fuente principal de problemas en la mayoría de dichos proyectos: requisitos inadecuados, cambios en los requisitos durante el ciclo de vida, requisitos no bien comprendidos y requisitos incompletos.

En 1987, el Departamento de Defensa de los Estados Unidos generó un informe [Brooks 87b] sobre los problemas de calidad y productividad en el desarrollo de software militar, que abrió las puertas para la investigación de la llamada “crisis del software”, ya reconocida desde fines de la década del 60 [Naur 68]. Paralelamente, acaecieron en la práctica industrial fracasos paradigmáticos en proyectos de gran envergadura, tales como el Sistema de Control del Tráfico Aéreo de la Administración Federal de Aviación [Gibbs 94] [Stix 94], el Sistema de Administración de Equipaje del Nuevo Aeropuerto Internacional de Denver [GAO 94], el Sistema de Ambulancias de Londres [Finkelstein 96], el Sistema de tiempo real de Paramax System Corporation [Lindstrom 93] y otros. Algunos de los casos más recientes que se pueden mencionar son: el Sistema de Reservas del Ferrocarril Nacional francés (dos días fuera de uso por defectos en la actualización de un programa, 2004), el Sistema de Gestión de Energía de General Electric (causó un apagón en el noreste de Estados Unidos y parte de Canadá, debido a un error de

programación en el sistema de gestión y monitoreo de energía de un proveedor que provocó la caída del resto de sistemas que no preveían esta falla, 2003), el software de la Estación Espacial Internacional (varios problemas tales como la operación de un brazo del robot de la estación, 2001), el programa de simulación del puente Millenium en Londres (falló debido a una incorrecta estimación de la cantidad de peatones, 2000), entre otros muchos.

En respuesta a la baja calidad de los productos de software con altos costos de corrección y mantenimiento, muchos desarrolladores de software e investigadores se han planteado la necesidad de profundizar en el proceso previo al diseño del software. Sus investigaciones y prácticas profesionales han apuntado a garantizar la producción de software de alta calidad y bajo costo basándose en el establecimiento de un punto de partida de alta confiabilidad, mediante la detección de errores lo más tempranamente posible. Es necesario entonces contar con procesos de producción de requisitos confiables.

En esta sub-sección, se ha dado cuenta de las consecuencias que producen los defectos en los requisitos, cabe mencionar a continuación las fuentes de dichos defectos. El reconocimiento de los defectos en los requisitos mediante el estudio de sus causas y efectos ha permitido, en los últimos años, el establecimiento de procesos de definición de requisitos atentos a esta grave circunstancia.

1.1.2. Errores en el proceso de desarrollo de software

Habitualmente cuando se hace referencia a errores en los requisitos de los sistemas se desliza en forma tácita la idea de que el profesional de software o el proceso de captura han sido el origen de los mismos. Esta suposición, si bien muy difundida, probablemente sea parcialmente falsa ya que muchos de los errores, discrepancias y omisiones que aparecen en los requisitos y naturalmente se propagan al sistema final, ya existían en la fuente de información. En realidad existe una responsabilidad compartida, ya que en los casos en que el defecto está empotrado en la fuente de información, el ingeniero de requisitos tiene algún grado de responsabilidad al no ponerlo en evidencia, especialmente en relación con las discrepancias y omisiones.

Mizuno en [Mizuno 83] propuso un modelo para la catarata de errores que se presenta en la Figura 1-1. Si bien este gráfico es notoriamente ilustrativo para comprender el problema de la propagación y la ampliación de los errores a lo largo del proceso de desarrollo de software, contiene varias simplificaciones excesivas que enmascaran algunos aspectos relevantes de la problemática de la Ingeniería de Requisitos.

El punto más importante parece ser la visión del problema real como una entidad fuente de todo saber y verdad. No existe un problema real perfecto que constituya la esencia químicamente pura de las necesidades del UdeD y que determine unívocamente los requisitos del sistema de software. Por el contrario, la dificultad realmente reside en que la visión que las organizaciones tienen de ellas mismas es incompleta o incoherente [Godel 62]. Peor aún las

fuentes de información introducen sus propias contradicciones, sesgos, errores y extemporalidades que obscurecen un poco más la percepción de las necesidades del negocio[2].

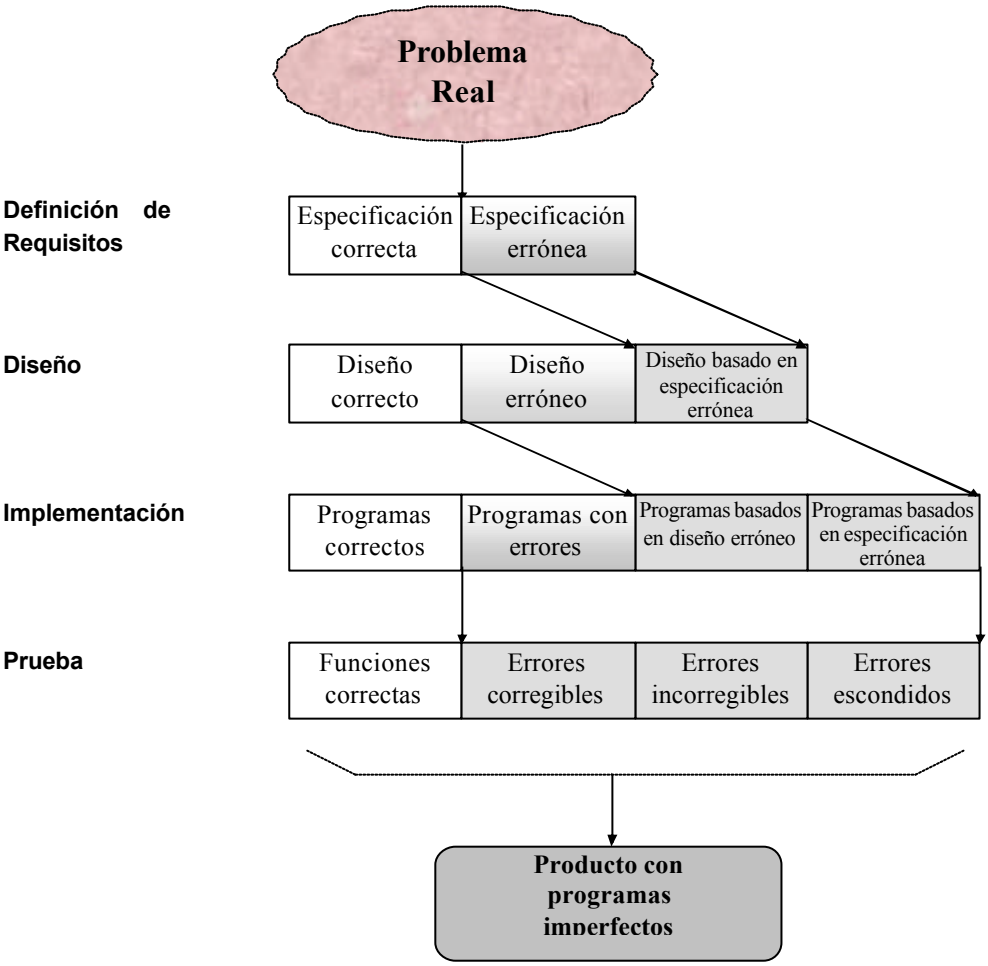


Figura 1-1. Catarata de Errores según el Modelo Mizuno [Mizuno 83]

El meta-modelo conceptual, que supone un "problema real" perfecto, asigna además todas las dificultades a:

i) la inhabilidad de las fuentes de información para revelar el conocimiento profundo, y
ii) la inhabilidad del ingeniero de requisitos para capturar la información relevante y libre de omisiones.

Esta visión ha entronizado el concepto de la captura de requisitos, ya que lo único que habría que hacer es "descubrir" la verdad. De esta manera los requisitos se han reducido a un conjunto de "pepitas de oro" y por lo tanto la tarea a emprender es tamizar la grava para encontrarlos.

El "problema real" es un agregado de certezas, contradicciones, dudas y

[2] En [Jacobson 99] se pone de manifiesto que una complicación en la captura de requisitos son los usuarios por ser éstos una "fuente imperfecta de información".

aspectos no planteados. En otras palabras, sí hay información directamente transformable en requisitos del sistema de software, pero hay mucho más que eso. La tarea del ingeniero de requisitos es mucho más compleja que meramente sonsacar los requisitos, también debe colaborar con el esclarecimiento de las dudas, debe detectar las contradicciones y visualizar las circunstancias no consideradas, entre otras. Adicionalmente, el ingeniero de requisitos contribuye con propuestas de servicios del sistema que en algunos casos complementan la información adquirida y en algunos otros ofrece algún antagonismo con la misma.

Los requerimientos, pedidos y demandas de los usuarios son idealmente realizados en el contexto del objetivo de obtener una mejora en los procesos del negocio. Se dice "idealmente" porque pueden insertarse objetivos personales no claramente explicitados o eventualmente cuidadosamente ocultos. Por el otro lado, las propuestas del ingeniero de requisitos están guiadas por la aspiración de facilitar el desarrollo del artefacto de software, ya sea simplificando el sistema o dotándolo de propiedades que lo acerquen a modelos conceptuales preconcebidos.

En la Figura 1-2, se presenta un esquema que extiende el modelo de Mizuno, visualizando el problema real como un agregado de certezas, contradicciones, dudas y aspectos no considerados. En este esquema además se representa el hecho de que las fuentes de información aportan su propia distorsión dificultando, en alguna medida, aún más el proceso de definición de los requisitos del sistema de software, proceso que por otro lado, como se visualiza en la figura, se descompone en tres sub-etapas: elicitación, modelado y especificación[3]. Por razones de espacio, no se ha graficado en la Figura 1-2 la cascada de errores que se arrastran en las siguientes etapas de diseño, implementación y prueba.

La Figura 1-2 fue confeccionada siguiendo el estilo de la figura original de Mizuno, por lo que conserva las mismas simplificaciones que la original, excepto las ya indicadas modificaciones en la visión del problema real. En ese sentido, el esquema no refleja que sobre los requisitos erróneos se puedan además acumular errores de diseño (sólo se cometen errores de diseño sobre los requisitos correctos) y que los errores de programación se concentran en los fragmentos del sistema correctamente diseñados. Por otra parte, en ambas figuras también se ignora que una actividad de validación forzaría, con un alto costo, la corrección de los "errores incorregibles"; así como descubriría en mayor o menor grado los "errores escondidos", también con altos costos de corrección.

Como se ha mencionado arriba, la tarea del ingeniero de requisitos no es simple, debe tamizar la grava en el UdeD para encontrar información relevante, certera y lo más completa posible, para lo cual debe contar con técnicas y herramientas adecuadas, saber lidiar con personas a distintos niveles de la organización, y seguir los pasos necesarios que le permitan asegurar el mejor resultado. Es decir, el ingeniero de requisitos debe adquirir

[3] La especificación puede ser vista como parte del modelado, pero se ha presentado por separado para mostrar más en detalle la cascada de errores.

habilidades y conocimientos que trascienden el campo tecnológico para entrar en contacto con otras ciencias que le den soporte integral a su actividad.

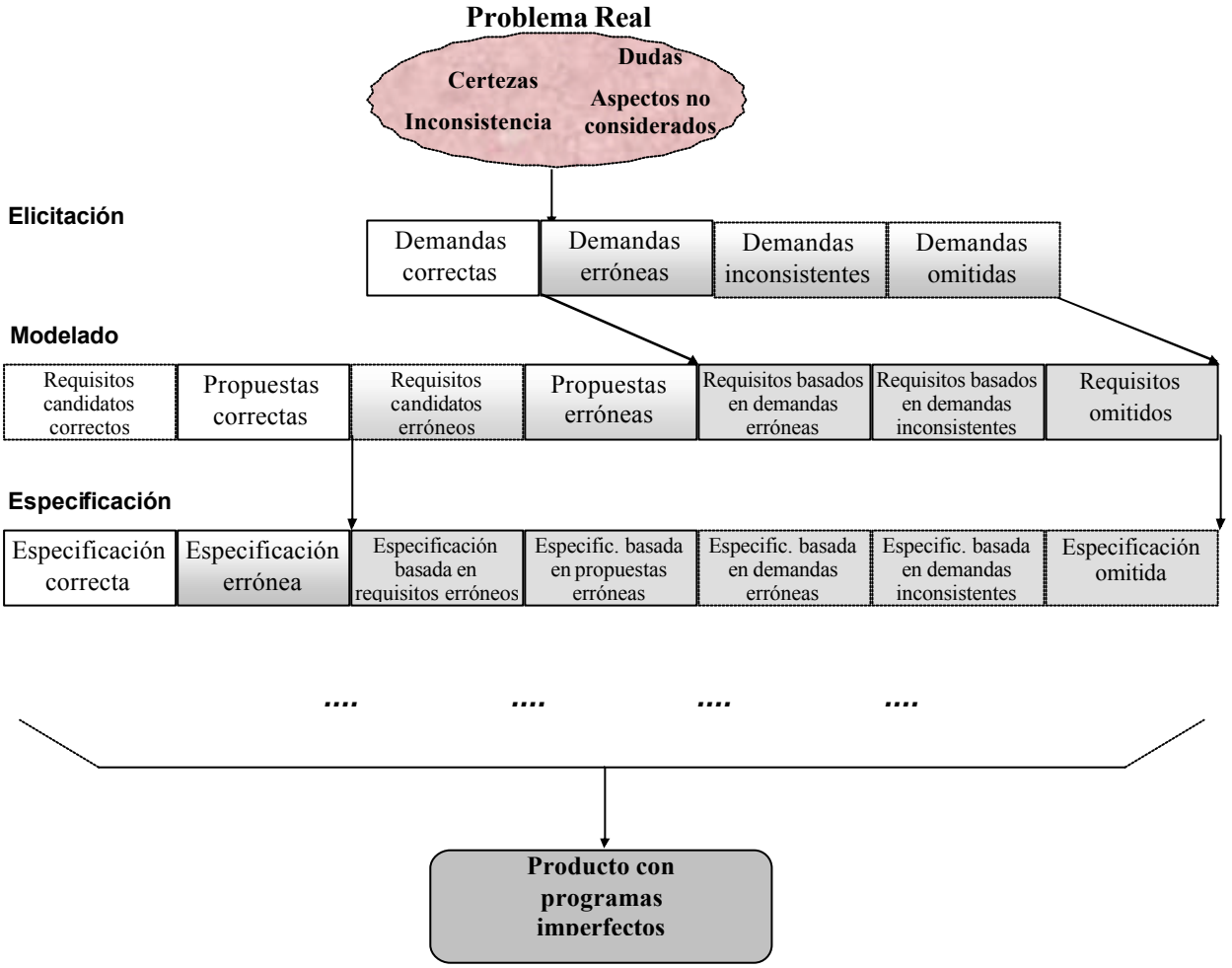


Figura 1-2. Catarata de Errores extendiendo el Modelo Mizuno

1.1.3. La Ingeniería de Requisitos y la Ingeniería de Software

La Ingeniería de Software (término acuñado por primera vez por un grupo de estudio de la NATO en 1968 [Naur 68]) provee un enfoque sistematizado para el desarrollo, operación, mantenimiento y retiro (desmantelamiento) del software [IEEE Std 610.12-1990]. Por lo tanto, como toda disciplina de la ingeniería, debe contar con las siguientes características: tecnologías bien comprendidas, procesos bien definidos, resultado predecible de las etapas del proceso y pasos del proceso repetibles. Se trata de una aspiración a la que tanto la Ingeniería de Software como la Ingeniería de Requisitos se están aproximando día a día, aún cuando Fred Brooks [Brooks 87] haya anunciado que nunca se encontrará la bala de plata: «*Ninguna mejora importante en el área de la ingeniería de software aparecerá nunca*». Dan Berry coincide con Brooks, pues luego de mostrar la frustración que causa el

desarrollo de software ya sea desde la aplicación de IR hasta XP, sugiere que tal vez el desarrollo de software sea un arte, independientemente de la cantidad de sistematización lograda en los procesos y métodos elaborados [Berry 02]. Pese a esto, la corriente principal de la Ingeniería de Software procura avanzar y hay mucho terreno para ello, y esto se evidencia en la cantidad y calidad de los trabajos de investigación que se están realizando en esta área.

Existe una gran diferencia entre el producto a construir por la ingeniería de software respecto a otras ingenierías. En casi todas las ramas de la ingeniería se genera habitualmente un bosquejo / maqueta / plano / diagrama u otro elemento que representa el artefacto a construir y, a través de él, se planifica la tarea de construcción y se asegura que responda a las expectativas de los clientes. Sin embargo, una dificultad especial que se le presenta a la ingeniería de software es justamente la representación del producto a construir, pues dicho producto (el software) es en sí mismo una representación del mundo real, entonces los documentos y/o modelos que describen el software a construir son una "representación de una representación" a ser creada [Kovitz 02]. Esto implica la necesidad de prestar especial atención a la generación de esos documentos y/o modelos que representarán a ese artefacto, lo que indudablemente requiere un considerable esfuerzo y tiempo de elaboración, para que sean precisos, correctos y lo más completos posibles. Esta es la tarea básica de la Ingeniería de Requisitos como pilar del desarrollo de software y, como tal, debe considerarse en el contexto de la Ingeniería de Software y de la Ingeniería de Sistemas [Kaindl 02].

Por tanto, se debe observar que la Ingeniería de Software se basa en dos conceptos fundamentales: *producto* [Gunther 78] y *sistema*. Es decir, tiene simultáneamente tanto un enfoque ingenieril como un enfoque sistémico. La función de la ingeniería es procurar productos de mejor calidad dentro de un costo compatible con esa calidad, optimizando costos. Pero además, dado que el producto es un sistema debe tener en consideración las características que presenta todo sistema: propósito, globalidad, entropía y homeostasis, de acuerdo a la Teoría General de Sistemas [von Bertalanffy 68].

La Ingeniería de Software tiene dos vertientes, una mejorar el proceso de producción de software con técnicas de desarrollo, y otra mejorar la gestión del proceso, y es en esta segunda parte donde fundamentalmente cabe la Ingeniería de Requisitos pues tiene más que ver con lo humano que con lo tecnológico. Se debe enfatizar que el proceso de desarrollo de software no es solamente una actividad técnica pues se lleva a cabo en un contexto social donde todos los involucrados participan [Leite 94] [Maté 05].

La Ingeniería de Requisitos es una disciplina que procura sistematizar el proceso de definición de requisitos. Es decir, apunta a mejorar la forma en que se comprenden y definen los servicios que deberán prestar los sistemas de software. La complejidad de los problemas a resolver exige que se preste más atención a su correcto entendimiento antes de comprometer una solución. La IR cubre todas las actividades involucradas en descubrir, modelar, analizar y mantener un conjunto de requisitos para un sistema de software.

Aunque el problema de cómo establecer sistemas no es nuevo, sólo recientemente la comunidad científica de computación, principalmente la comunidad de ingeniería de software, comenzó a prestar especial atención a la tarea de definición de los servicios que deben prestar los sistemas. El término "Ingeniería de Requisitos" surgió (formalmente publicado) recién en Enero de 1977 en IEEE Transactions on Software Engineering, donde figuran los siguientes artículos acuñando el término:

- "*A Requirements Engineering Methodology for Real-Time Processing Requirements*", Alford, M., IEEE-TSE, January 1977.
- "*An Extendable Approach to Computer-Aided Software Requirements Engineering*", Bell, T., Bixler, D., Dyer, M., IEEE-TSE, January 1977.

La primera conferencia internacional sobre este tema fue en 1993 (RE'93) y en 1995 se realizó la primera reunión del grupo de trabajo IFIP 2.9 (Ingeniería de Requisitos de Software).

La Ingeniería de Software ha estado prestando más atención al modelado e implementación (codificación) de productos, proveyendo métodos, herramientas y procedimientos para tal fin, mientras que la Ingeniería de Requisitos comenzó examinando los aspectos del establecimiento de requisitos y su intrínseca dependencia de los aspectos sociales que rodean la construcción del software y que rodearán su operación.

La IR tiene una interacción muy fuerte con aquellos que demandan un producto de software. Por lo tanto, el proceso de la IR es un esfuerzo cooperativo que involucra a usuarios, clientes, ingenieros de software, consultores, expertos, denominados en la literatura "stakeholders", siendo el término que se emplea en castellano "involucrados"[4].

La IR debe entonces achicar la brecha entre el usuario que tiene necesidades y requiere soluciones a sus problemas, pero que no tiene los conocimientos para especificar los requisitos, y el ingeniero de software que sí tiene dichos conocimientos técnicos pero desconoce el mundo del usuario con sus problemas y necesidades. De lo dicho, surgen dos aspectos ineludibles en un proceso de IR exitoso:

i) establecer una buena comunicación entre ambas partes, y
ii) comprender cabalmente el mundo del usuario.

Lo primero sólo es posible lograrlo cuando el ingeniero de software comprende el vocabulario que maneja el usuario. La postura inversa es casi

[4] Se entiende por *clientes* a los involucrados en el UdeD que de alguna manera se constituyen en dueños del sistema, es decir aquellos que demandan el sistema de software y *usuarios* a aquellos que efectivamente deban interactuar con el mismo. Naturalmente que un *cliente* puede ser simultáneamente un *usuario* y frecuentemente lo es; inclusive éstos no son roles estáticos, por el contrario pueden interactuar con el ingeniero de requisitos cambiando su punto de vista de *cliente* a *usuario* y viceversa en forma notoriamente dinámica. Se puede decir que el punto de vista del *cliente* tiene un sesgo más fuerte hacia los objetivos del sistema a ser desarrollado, y que el del *usuario* está más dirigido a los detalles concretos de su trabajo diario.

insostenible: no es esperable que el usuario posea conocimiento previo o sea capacitado específicamente para expresarse utilizando el vocabulario técnico o para interpretar los documentos técnicos del ingeniero de software.

Para lo segundo, el ingeniero de software requiere utilizar técnicas específicas que ayuden a obtener información de ese mundo (técnicas de elicitación) y que faciliten determinar si la comprensión lograda es correcta (técnicas de V&V). Cabe destacar que cuando la fuente de información son las personas, éstas muchas veces no saben o no pueden o no quieren expresar claramente sus necesidades, o lo que es peor aún, pueden ni siquiera ser conscientes de ellas.

En resumen, la necesidad de asegurar un buen entendimiento entre los ingenieros de software y los usuarios (en general, entre todos los involucrados) ha conducido al desarrollo de métodos que permitan la colaboración entre todos los participantes del proceso de definición de requisitos. Los ingenieros de requisitos deben comprender, modelar y analizar el UdeD donde el software operará y los usuarios deben confirmar que la visión de los ingenieros es correcta.

Existen además otras dificultades que hacen que el desarrollo de software no sea una actividad lineal y sencilla, que alcance con sólo concentrarse inicialmente en la definición de requisitos para luego dedicarse a diseñar e implementar la solución. Múltiples causas provocan continuos cambios en el software, los que con frecuencia deben atenderse casi inmediatamente a su aparición. La naturaleza de estos cambios corresponde en su gran mayoría a requisitos cambiantes.

1.1.4. Evolución de los Requisitos

La Ingeniería de Software desde sus comienzos ha propuesto métodos para facilitar los cambios en el software [Lehman 85] [Lehman 91], desde los métodos estructurados, ocultando información en módulos hasta los métodos orientados a objetos encapsulando comportamiento en clases. Los cambios no son algo ocasionales sino que, muy por el contrario, ocurren constantemente durante todo el ciclo de vida del software. Como bien menciona Leite [Leite 97b]: “el cambio es una propiedad intrínseca al software”, y por lo tanto merece una consideración especial en el proceso de desarrollo: la administración de dichos cambios. Desde la perspectiva de la IR, esta actividad se denomina: la gestión de requisitos.

Los requisitos evolucionan registrando los cambios que el UdeD impone al software. Estadísticas realizadas marcan que el 50% o más de los requisitos van a cambiar antes que el sistema de software se ponga en operación [Kotonya 98]. Es por esta razón que se pone énfasis en especificaciones dinámicas de requisitos precisos y claros que acompañen los cambios del UdeD y que permitan el seguimiento de los mismos desde cualquier punto del proceso de producción del software hasta sus orígenes.

Pero debe cuestionarse cuál es el origen de estos cambios en el software. ¿Son estos cambios realmente genuinos, es decir, surgen como consecuencia de cambios en el UdeD, o provienen de requisitos descubiertos tardíamente? Es frecuente que algunos requisitos se descubran parcialmente o no salgan a la luz hasta avanzado el desarrollo, cuando los diseñadores o inclusive los programadores deben realizar tareas a muy bajo nivel de detalle y se chocan con ambigüedades, contradicciones u omisiones. En estos casos, la esencia del cambio no corresponde a un mundo en evolución sino a requisitos no descubiertos oportunamente. Con respecto a los cambios "reales" en el UdeD, no sólo se deben a cambios externos o internos a la organización como consecuencia de factores políticos, económicos, legales, sociales o tecnológicos, sino muchas veces a un cambio en las expectativas de los mismos clientes y usuarios una vez que ven más claramente las posibilidades que permitirá el sistema de software. Lehman [Lehman 80] va más allá expresando que un sistema una vez instalado se convierte en parte del UdeD alterándolo y, por ende, él mismo (el software) altera sus propios requisitos.

En general, las estadísticas realizadas sobre la evolución de los requisitos no discriminan adecuadamente el origen de los cambios en el desarrollo del software, y simplemente lo adjudican a la evolución del UdeD [Yeh 90] [Kotonya 96] [Leite 97] [Madhavji 97] [Goedicke 99]. Lientz & Swanson [Lientz 78] determinaron que los cambios en el software durante su mantenimiento se debían en un 80% a cambios en los requisitos y el restante 20% a correcciones de errores. Sin embargo, no se discrimina en estos valores cuántas de las correcciones de errores se debían a requisitos con defectos ni cuántos de los cambios en los requisitos eran por la naturaleza evolutiva del UdeD o por defectos en los requisitos (requisitos descubiertos tardíamente o requisitos mal interpretados). Un estudio más reciente sobre mantenimiento de software [Pigoski 96] informa que alrededor del 55% de los pedidos de cambio corresponden a requisitos nuevos o cambios en requisitos por mejoras, mientras que un 25% son cambios técnicos por adaptaciones.

Desde ya que existen UdeD donde predomina una evolución constante, tal es el caso por ejemplo del dominio de telecomunicaciones [Zave 01]. También es el caso de desarrollos de sistemas grandes, complejos donde es inevitable que haya múltiples versiones que derivan tanto de cambios planeados por mejoras como de cambios no planeados por evolución del UdeD [Faulk 96]. Por otro lado, el descubrimiento tardío de requisitos se debe en muchos casos a procesos de producción "débiles" con pocas guías y poca disciplina, donde se requiere la generación rápida del artefacto de software y los requisitos se descubren a medida que se implementa parcialmente el software, tal es el caso en el dominio del comercio electrónico donde hay una fuerte competencia [Carter 01]. Es muy común también que el descubrimiento tardío corresponda a requisitos no funcionales (RNF) (ver sub-sección 2.6.4), cuando éstos no son atendidos adecuadamente durante la definición de requisitos [Cysneiros 99], un RNF no capturado oportunamente puede obligar a muchos cambios en otros requisitos tanto funcionales como no funcionales. Faulk en [Faulk 96] denomina a estos cambios en los requisitos: "dificultades accidentales", causadas por una mala administración, elicitación o definición de requisitos, y son estas dificultades las que son más fáciles de evitar con buenas

prácticas y procesos en la IR.

En un estudio de campo sobre 10 organizaciones realizado por [Lubars 93] se reporta sobre la frecuencia con que cambian los requisitos durante el proceso de desarrollo y se distinguen dos tipos de cambios: i) cambios en los requisitos mismos, y ii) cambios en la forma en que son expresados. Los primeros son cambios solicitados por los clientes y usuarios debido a cambios en el medio ambiente, identificando diversas razones. Mientras que los segundos son cambios solicitados por los desarrolladores debido a la detección de ambigüedades.

Entonces, la frase *«La evolución de los requisitos es debido a un mundo en constante cambio»* tiene menos peso de lo que uno presupone una vez estudiadas las causas que promueven cambios en los requisitos.

De lo mencionado hasta ahora, cabe destacar dos cuestiones:

i) La visión del Modelo de Mizuno (Figura 1-1) y la visión extendida del mismo (Figura 1-2) es estática. Ambas consideran que los procesos del negocio no cambian durante el período de desarrollo del software.

ii) La visión muy difundida de la evolución de los requisitos se basa en un único componente: el cambio del UdeD, pero prácticamente omite un segundo componente: los requisitos descubiertos o comprendidos tardíamente. Es obvio que este último componente produce también modificaciones en los requisitos, y lo hace con mucha mayor frecuencia de lo mencionado en la literatura[5].

En la Figura 1-3 se muestra la incorporación de la noción de evolución de requisitos a la Figura 1-2, donde se muestra un eje relativo al tiempo de desarrollo del software y otro relativo al tiempo de los cambios durante el desarrollo mismo.

El proceso de la IR (ver capítulo 2) tiene sentido en organizaciones con una estabilidad aceptable en sus procesos del negocio[6]. En organizaciones con estrategias o procesos del negocio inestables a muy inestables, hay que pensar en métodos ágiles de desarrollo de software, pues cuál es el sentido de modelar requisitos que cambian más rápido que el tiempo necesario para documentarlos.

Los sistemas de software son un factor de estabilización de los procedimientos de la organización, por lo tanto, es natural que el proceso de desarrollo del software tenga un cierto grado de conflicto con la tendencia a la variación de estas organizaciones. En organizaciones con cambios constantes, los sistemas de software se ven sumidos en un mantenimiento permanente e incremental a medida que nuevos componentes de software se van instalando.

[5] Ya en 1994 Michael Jackson afirmaba que sobre los requisitos se saben dos cosas: 1) van a cambiar y 2) van a ser mal comprendidos [Jackson 94].

[6] Se dice que un dominio de aplicación es estable cuando el conocimiento sobre los procesos del negocio y sobre las expectativas de los usuarios es claro, y generalmente existe homogeneidad en los intereses de los usuarios [Prahalad 02].

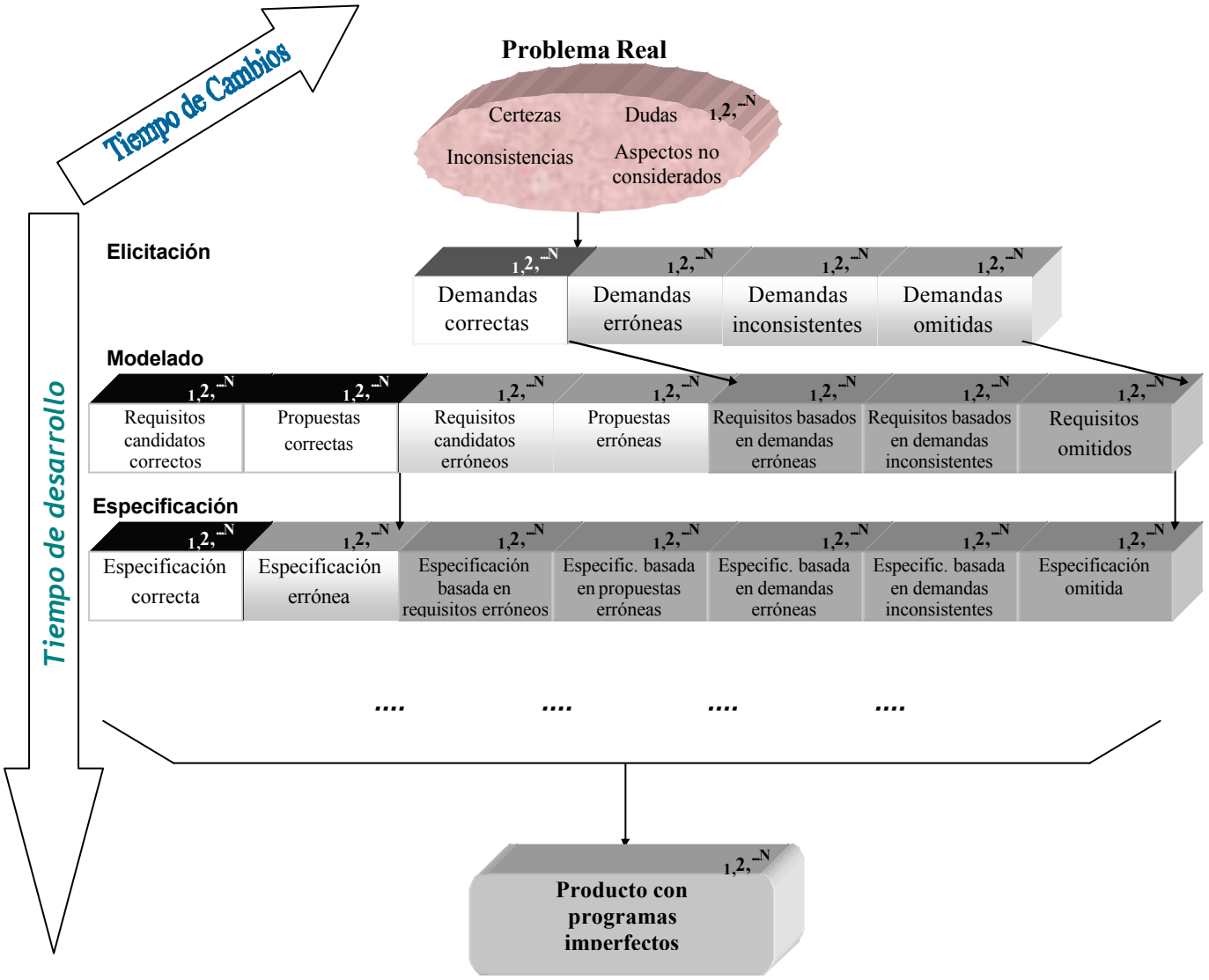


Figura 1-3. Evolución de los Errores + Evolución de los Requisitos

La evolución de los errores y la evolución de los requisitos son dos características del desarrollo de software estrechamente vinculadas. Belady & Lehman [Belady 76] construyeron un modelo que representa la evolución de los errores en un sistema de software a lo largo del tiempo, en función de las versiones del software que se generan para corregir errores en los programas y actualizar / mejorar la funcionalidad del mismo (Figura 1-4). El punto mínimo de la curva representa la versión de software con la menor cantidad de errores, pasado el cual la curva asciende rápidamente, pues cualquier cambio a realizar en el software se torna cada vez más dificultoso debido a un paulatino y constante desmejoramiento en su arquitectura, lo cual a su vez facilita la gestación de nuevos errores. Los métodos de la Ingeniería de Software deberían tender a que la parte ascendente de la curva de Belady-Lehman posterior al punto mínimo sea lo más suave posible postergando el aumento de la entropía del sistema. ¿Qué debería entonces aportar la IR para que esto ocurra? En la sub-sección siguiente se intenta dar una respuesta a ello.

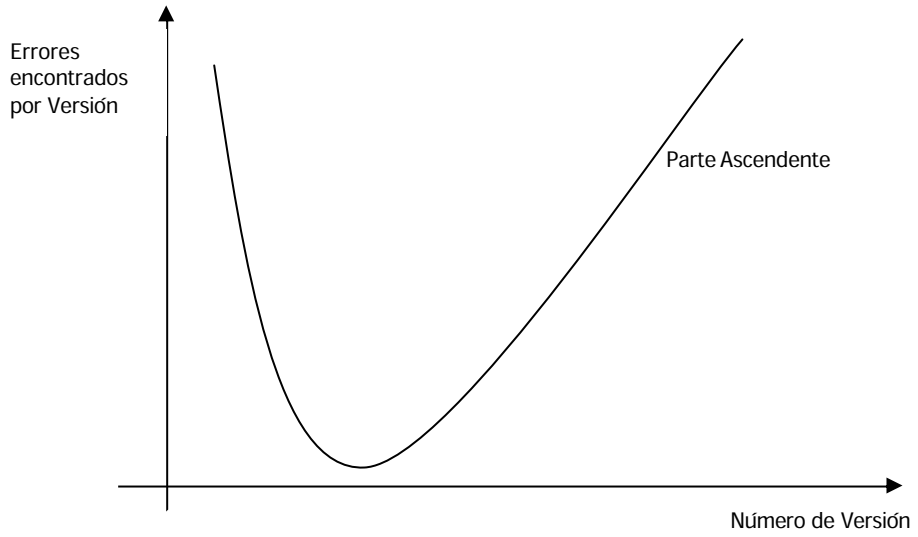


Figura 1-4. Gráfico Belady-Lehman

1.1.5. Reflexiones

Dados los graves problemas que los errores en los requisitos ocasionan sobre el producto final (el sistema de software) y durante el proceso de producción del mismo, es que diversos centros de estudio, organismos gubernamentales y otras organizaciones se han dedicado en la última década a la investigación en el área de los requisitos de los sistemas de software.

En tal sentido, se han propuesto diversos enfoques en el tratamiento de requisitos, estableciendo nuevos procesos, técnicas y métodos de producción de requisitos (que se detallan en el capítulo 2), creando herramientas de soporte a dichos procesos, fijando estándares en los procesos y productos, y estableciendo procesos para garantizar la calidad de los requisitos, procesos que permitan la evolución de los requisitos y su adecuada gestión durante todo el ciclo de vida del software. En el Apéndice A, se enumeran algunos ejemplos importantes de estas iniciativas.

En resumen, los puntos claves que debe considerar un proceso de Ingeniería de Requisitos son:

i) Involucrar a los clientes y usuarios en el proceso. Es decir, establecer la participación activa y constante de los clientes y usuarios. Esto implica entablar una buena comunicación con ellos para comprender cabalmente sus necesidades.
ii) Asegurar la calidad de los requisitos. Esto involucra utilizar técnicas adecuadas de V&V, las cuales deben formar parte de las actividades continuas del proceso de la IR y no como hitos estáticos. Por otro lado, la calidad de los requisitos debe conservarse a lo largo de todo el ciclo de vida del software.
iii) Considerar la evolución de los requisitos durante el ciclo de vida del software. El proceso de desarrollo debe acompañar esta evolución. Esto involucra una adecuada gestión de requisitos, asegurando la

trazabilidad de los mismos para facilitar la realización de los cambios en los modelos y en el software, evitando la introducción de nuevos defectos a consecuencia de los cambios.

iv) Considerar aspectos sociales, organizacionales, políticos y legales del ambiente donde el software operará. Esto impone una inmersión de los desarrolladores en dicho ambiente y el uso de técnicas de modelado organizacional y de técnicas de recolección de datos apropiadas al problema bajo estudio.

v) Procurar la completitud de los requisitos[7]. Esto es por un lado, un aspecto de la calidad y por otro lado, requiere de una comprensión acabada del UdeD, ambos aspectos tratados en los puntos anteriores.

Todos los puntos arriba mencionados son aportes de la IR que benefician manifiestamente el proceso de desarrollo de software, dado que las actividades de la IR son parte constitutiva de él. A pesar de los beneficios que conlleva la práctica de la IR, se perciben obstáculos para la transferencia de tecnología entre la investigación en el marco de la IR y su práctica profesional, obstáculos tales como la falta de educación en esta disciplina, la falta de estándares en lenguajes para la especificación de requisitos, la falta de herramientas integradas al proceso completo de desarrollo, la falta de apoyo por parte de las organizaciones para adoptar nuevos métodos de la IR, la brecha existente entre el análisis del negocio y el análisis de los requisitos, entre otros [KaindI 02]. Un estudio reciente sobre la práctica de la IR [Neill 03] reveló que un 52% de los encuestados consideran suficiente el esfuerzo realizado por sus organizaciones en IR, mientras que un 29% lo considera insuficiente.

1.2. Objetivos y Trabajo realizado

Dentro del marco descripto en el punto anterior (sub-sección 1.1.3) varios autores han propuesto modelos para formalizar la obtención de requisitos, tales como [Jacobson 92], [Reubinstein 91], [Bubenko 93], [Loucopoulos 95] y otros. El modelo propuesto por el Prof. Julio Leite está basado en un andarivel denominado *Client-oriented Requirements Baseline* [Leite 95], que utiliza una estructura dinámica basada en el dominio del problema.

Esta tesis se basará en el uso de dos modelos dentro del *Client-oriented Requirements Baseline*: el Léxico Extendido del Lenguaje [Leite 93] y Escenarios [Leite 97]. El primer modelo permite comprender el vocabulario de la aplicación, mediante una descripción de la denotación y connotación de cada

[7] Falacia de la completitud [Leite 94]: en un mundo real de gran complejidad, se aspira a obtener un conjunto de requisitos lo más completo posible. *«El proceso de definir requisitos es inherentemente incompleto, teniendo en vista la gran complejidad del mundo real. Es obvio, por lo tanto, que siempre estaremos procurando tener requisitos los más completos posibles. La imposibilidad de completitud está relacionada con la afirmación de Brooks [Brooks 87] sobre la bala de plata y la característica de homeostasis de los sistemas, o sea la característica evolutiva del software, que procura adaptarse a su macrosistema»* [Leite 01b].

palabra / frase utilizada en el UdeD. Por otro lado, el segundo modelo, Escenarios, tiene como objetivo comprender el problema, mediante la descripción de situaciones del UdeD. Cabe destacar que, Escenarios es un modelo de representación ampliamente difundido en la literatura [Potts 95], [Carroll 95], [Sutcliffe 98], [Rolland 98], [Rolland 98b] y utilizado exitosamente en la práctica [Weidenhaupt 98].

Ambos modelos propuestos se caracterizan por tener una representación textual basada en lenguaje natural. El registro de los requisitos de sistemas informáticos mediante el uso del lenguaje natural es sumamente atractivo ya que permite involucrar a la totalidad de los agentes intervinientes en el proceso de desarrollo sin las barreras de conocimiento clásicas que se producen cuando se utilizan otras formas de representación. Se ha comprobado que el uso de una estructura ligeramente formalizada del lenguaje natural, como es el uso de los Escenarios y del Léxico Extendido del Lenguaje, no introduce una dificultad apreciable cuando es leída por personas no especializadas en el desarrollo de aplicaciones de software.

Estadísticas [Weidenhaupt 98] acerca del uso de escenarios en la realidad productiva han comprobado que los mismos requieren esfuerzos significativos para su administración y que su uso es un "arte" más que un proceso formal. Rolland y Ben Achour en [Rolland 98c, p.126] expresan la falta de guías y recomendaciones en la práctica de la IR basada en escenarios. Asimismo, Sutcliffe en [Sutcliffe 97] enfatiza la efectividad de los escenarios en el desarrollo de software pero puntualiza:

> *«We have little understanding about how scenarios should be constructed, little hard evidence about their effectiveness and even less idea about why they work»*
>
> [Sutcliffe 97]

En esta tesis se formalizará el uso de escenarios en la fase de Ingeniería de Requisitos, permitiendo establecer técnicas para la verificación de los mismos. Esta tesis es la continuación de la investigación realizada cuyos resultados se exponen en [Hadad 97], [Leite 97] y [Doorn 98].

Se propone una estrategia para la definición de requisitos, que consiste básicamente en: primero, comprender el UdeD presente, para ello se construyen el Léxico y los Escenarios que representan situaciones actuales; luego, comprender el UdeD futuro, para ello se construyen Escenarios que representan situaciones del futuro, y finalmente, especificar los requisitos del sistema de software basándose en el conocimiento adquirido y registrado en las etapas previas. Esta tesis se centra en las tres primeras etapas, para lo cual se han perfeccionado y detallado heurísticas generales existentes para la construcción del Léxico y se han desarrollado heurísticas detalladas para la construcción de escenarios actuales y escenarios futuros. Por último, se presenta una aproximación inicial para explicitar requisitos a partir de los escenarios construidos.

Se han desarrollado heurísticas precisas en la construcción del Léxico Extendido del Lenguaje, que involucra información que se distingue desde el punto de vista de la vigencia en: lo *actual* (es decir, lo que *es / ocurre* realmente en el UdeD) y lo *formal* (es decir, lo que *debe ser / ocurrir* pero no siempre ocurre). Las diversas fuentes de información del UdeD a menudo proveen datos que involucran más de un punto de vista, en muchos casos con un sesgo a algún punto de vista en particular; todos estos tipos de información deben registrarse adecuadamente en los modelos de representación, alentando en el ingeniero de requisitos la captación y reconocimiento de las mismas. Esto sólo es posible a través de modelos y técnicas preparados para tal fin.

Asimismo se propone desarrollar dos conjuntos de escenarios del problema bajo estudio. El primero de ellos será el conjunto de escenarios actuales del macrosistema y el segundo de "lo que se espera que sea en el futuro", denominado escenarios futuros. Los Escenarios Actuales registran las situaciones del UdeD del presente e integran en forma coordinada y armónica lo que ocurre en él y lo que debiera ocurrir formalmente. Por otro lado, los Escenarios Futuros modelan lo esperado o deseado respecto del futuro. Los Escenarios Futuros tienen una relación estrecha con los Escenarios Actuales ya que se refieren al mismo UdeD sobre el que se conjeturan evoluciones de las características del presente y se pretende introducir modificaciones / mejoras. Los Escenarios Futuros son de fundamental importancia pues permiten modelar no sólo las características del sistema de software a ser desarrollado sino también la influencia de éste sobre los restantes componentes del UdeD y el contexto esperado para el mismo.

El método básico de trabajo consistió en un refinamiento del proceso existente de construcción del Léxico, precisando su heurística, incorporando el punto de vista de la vigencia en la información y especificando un tratamiento de las ambigüedades en el vocabulario, mediante reconocimiento de sinónimos, tratamiento de homónimos e identificación de jerarquías de términos. Posteriormente, se especificó el proceso de construcción de escenarios actuales, usando la heurística de derivación formulada en [Hadad 97], incorporando mecanismos para organizar e integrar los escenarios y desarrollando un proceso de verificación de los mismos. Finalmente se realizó un análisis conceptual de las distintas estrategias para construir escenarios futuros. Una vez lograda la formalización, la misma se comprobó en casos de estudio de manera de evaluar tanto los aspectos formales como los aspectos pragmáticos para establecer con claridad la precisión de la información obtenida. A partir de esta evaluación, se identificó la mejor estrategia a seguir dependiendo del tipo de aplicación y, por otro lado, asegurar que los métodos propuestos sean a la vez correctos y utilizables en la práctica.

En el apéndice B, se enumera la producción científica lograda a través de diversas publicaciones y ponencias en congresos como resultado de la línea de investigación presentada.

En mayor o menor grado, el desarrollo de esta tesis, especialmente sus publicaciones intermedias, han influenciado el desarrollo de los trabajos que se mencionan en el apéndice C.

1.3. Estructura de la tesis

Esta tesis está organizada en 9 capítulos y un apéndice con 6 secciones. En el presente capítulo se ha descripto la motivación que ha llevado a la presentación de esta tesis y el plan desarrollado.

En el capítulo 2 se presenta el marco teórico de la tesis, que involucra los conceptos de proceso de la IR y métodos de la IR, la IR en los procesos de software, la diferenciación entre requisito, requerimiento y especificación, glosarios, escenarios y casos de uso y baseline de requisitos.

En el capítulo 3 se presenta una estrategia en la IR, los modelos que utiliza, la evolución de los mismos, y cómo utilizar la estrategia.

En el capítulo 4 se introduce un glosario particular, denominado LEL, su modelo de representación, el proceso de creación del LEL detallando heurísticas y finalmente, observaciones sobre la práctica de glosarios en la IR.

En el capítulo 5 se presenta el concepto de situación, un modelo de representación de escenario, la distinción entre escenario actual y escenario futuro, y observaciones respecto al modelo y a la distinción de escenarios.

En el capítulo 6 se presenta un enfoque middle-out en la construcción de escenarios actuales, sus actividades y heurísticas, un detalle del proceso de inspección de escenarios y su administración, observaciones respecto a otros enfoques de construcción de escenarios y observaciones respecto al proceso de inspección.

En el capítulo 7 se presenta un proceso de construcción de escenarios futuros, permitiendo la selección de tres estrategias para el modelado de los mismos según el grado de mejoras establecido para los procesos del negocio. Se muestran observaciones sobre otros enfoques que tienen en cuenta la administración de escenarios y objetivos.

En el capítulo 8 se presenta un proceso para poner en evidencia los requisitos empotrados en los escenarios futuros, con el fin de producir un SRS de ser necesario.

En el capítulo 9 se muestran futuros trabajos de investigación sobre la base de esta tesis que abarcan temas en el LEL, la relación entre escenarios y objetivos, la trazabilidad a modelos externos a requisitos, la relación entre escenarios y el diseño arquitectónico, el impacto de los escenarios futuros en la negociación y una forma de detectar los mejores usuarios para elicitar información.

Los apéndices abarcan desde información sobre diversos temas de la IR, otras tesis relacionadas con la presente, publicaciones integrantes de esta tesis, un caso de estudio que abarca todos los modelos de la estrategia, un

detalle sobre el modelo de trazas con dos ejemplos del meta-modelo, un ejemplo de aplicación del proceso de inspección de escenarios, y la relación entre el modelo de escenario y el modelo de caso de uso.

Capítulo 2
Ingeniería de Requisitos

«No part of the delivery process is as critical, nor as difficult - because requirements map the human world to the technological world. »

Jim Highsmith
Frase expuesta en la página de Ian F.Alexander:
http://easyweb.easynet.co.uk/~iany/consultancy/consultancy_welcome.htm

2.1. Consideraciones iniciales

El presente marco teórico ha sido construido con un punto de vista post-factum que no era el que se disponía al comenzar la tesis. La razón de esto se hace evidente naturalmente al mirar la cantidad y calidad de las publicaciones realizadas desde la primera aproximación a este tema.

En otras palabras, durante el desarrollo de esta tesis muchas novedades fueron introducidas [Mylopoulos 03] [Dorfman 97] en las estrategias de otros autores que se describen a continuación, inclusive algunos de los cambios en las estrategias de otros autores fueron consecuencia de publicaciones realizadas en el marco de esta tesis. Naturalmente, que las publicaciones nuevas de otros autores influyeron de manera significativa en el decurso de esta tesis. Un marco teórico de lo que "existía" antes de esta tesis hubiera dejado muchas cosas sin explicar y oscurecido la descripción del desarrollo de la misma. Por lo tanto, el presente marco teórico es lo más cercano posible al dominio actual de la Ingeniería de Requisitos.

2.2. El Proceso de la Ingeniería de Requisitos

2.2.1. Contexto del proceso de la IR

El contexto en el que se lleva a cabo el desarrollo de un software puede ser muy diverso, desde un proyecto interno de la organización hasta un desarrollo tercerizado o la compra de un COTS, o desde una actualización / ampliación de un sistema existente hasta un desarrollo de un producto completamente nuevo, o desde un proyecto de desarrollo a medida para un cliente específico hasta un proyecto de desarrollo dirigido a un mercado.

Todas estas variantes, adicionando factores como el tipo de aplicación, el tipo de organización cliente, los grupos humanos que componen la organización cliente, el grupo humano de desarrolladores, entre otros, obligan a la adaptación del proceso de desarrollo del software y más en particular al proceso de la IR, donde debe existir una fuerte interacción entre los grupos humanos involucrados.

Cuando se trata de un proyecto de desarrollo de aplicaciones a medida con proveedores y clientes pertenecientes a diferentes organizaciones, aparece claramente la existencia de un marco de trabajo para la IR que involucra la existencia de pedidos, ofertas (eventualmente de varios proveedores), órdenes de compra, plazos de entregas y eventualmente contratos con anexos técnicos donde se describe el producto a proveer. Aquí el cliente puede jugar un papel de fuerte exigencia hacia el proveedor.

Cuando se tiene que los desarrolladores son parte de la misma organización pero no dependen del área del cliente, hay una situación de pedidos y compromisos de desarrollo, eventualmente no respaldados por instrumentos legales pero con un grado importante de exigencia en el cumplimiento. Mientras que cuando los desarrolladores internos dependen del área del cliente, se le presta menos importancia al impacto de lo pedido y existe una exigencia más fuerte de cumplimiento tal como se enuncia el pedido y con menor posibilidad de negociación.

Un ejemplo muy particular es el de la generación del pliego para la licitación de un producto de software, donde la organización licitadora es cliente para la licitación, pero también puede ser la propia gestora del pliego (rol de proveedor), donde se requiere llevar a cabo un proceso de IR que concluirá con una definición de los requisitos del producto como parte esencial de dicho pliego.

Por otro lado, cuando el proyecto corresponde al desarrollo de un paquete de software para el mercado, no existe un cliente específico ni usuarios como principal fuente de información, y la empresa de software amortiza sus costos de desarrollo entre muchos clientes, siendo entonces el papel del desarrollador dominante en la relación que entabla con sus clientes.

De los ejemplos expuestos y de muchos otros casos, se observa que puede establecerse una posición dominante en la relación proveedor – cliente en un proyecto de software. Es, por ende, que el proceso de la IR debe adaptarse según el rol que juegan en el proyecto la parte cliente y la parte proveedora del software.

Cuando el cliente ejerce una posición dominante, la actividad del ingeniero de requisitos se puede desarrollar normalmente o no dependiendo del grado de reconocimiento que el cliente tenga de la misma. Es frecuente encontrarse que por ansiedades personales o institucionales, por situaciones de crisis, por problemas de competencia o por la imposición externa de plazos perentorios, un producto de software deba desarrollarse en plazos exiguos. En estos casos suele verse la actividad del ingeniero de requisitos como una pérdida de tiempo o un costo innecesario y por lo tanto se la dificulta o se la reduce a una mínima expresión. Esto provoca un impacto altamente negativo sobre el costo global de un sistema de software a lo largo de toda su vida, sin embargo estas situaciones siguen ocurriendo casi a diario.

Cuando el proveedor ejerce una posición dominante, el ingeniero de

requisitos no suele interactuar mucho con clientes o usuario concretos, sino con difusas nociones como usuarios hipotéticos y en alguna medida, los vendedores del producto, o eventualmente estadísticas y estudios de mercado se transforman en sus clientes. La definición de los requisitos de un producto de software se produce a partir de fuentes de información muy variadas, donde la implementación de estos requisitos puede originar modificaciones en las formas de trabajar de los futuros usuarios, pero intentando que sean consideradas favorablemente o en forma neutra por los mismos.

Naturalmente que es también frecuente encontrar casos de equilibrio, en las que el proveedor y el cliente concilian sus intereses logrando el cliente un producto muy apropiado para el proceso de su negocio y el desarrollador influenciar parcialmente en ese proceso del negocio para facilitar en algún sentido su trabajo. En estas situaciones tiene lugar una actividad que no es habitualmente muy reconocida, pero cuyo efecto práctico sobre las características del producto de software es visible e importante. Se trata de la negociación de requisitos. Los clientes y usuarios, implícita o explícitamente solicitan conductas o servicios del software basados en las características de su negocio, ignorando total o parcialmente los efectos que las mismas tendrán sobre los costos de desarrollo, calidad o mantenimiento posterior. Los ingenieros de requisitos suelen percibir algunas de esas solicitudes como arbitrarias o innecesarias, fundamentalmente porque visualizan cambios en el proceso del negocio que permitirían concebir un producto de software mejor, en términos técnicos. Entonces, los ingenieros de requisitos suelen hacer propuestas que aspiran a modificar, total o parcialmente, algunos de los servicios solicitados por los clientes. Esta negociación, adecuadamente canalizada, produce habitualmente beneficios significativos en la calidad de los requisitos y naturalmente luego en la calidad del software desarrollado.

En resumen, se puede concluir que el factor humano juega un papel de vital relevancia en el desarrollo de software, pudiendo incluso con frecuencia actuar como un obstáculo para resolver problemas técnicos. El factor humano [Saroka 02] no sólo involucra aspectos psicosociales y de comportamiento de ambos grupos humanos (clientes – desarrolladores) sino la cultura de la organización cliente, que desempeña un rol decisivo en el proceso de desarrollo del software (ver sub-sección 2.3.2). En [Coulin 05, p.47] se ha expuesto una lista importante de desafíos que presenta la IR, la gran mayoría asociados a los grupos humanos intervinientes y a la comunicación entre ellos.

2.2.2. Aspectos ético contractuales de la IR

Independientemente si los desarrolladores o proveedores de software, y los clientes y usuarios pertenecen o no a la misma organización, ambos grupos humanos tienen una cultura y una visión sobre cada posible proyecto de desarrollo de software diferente.

Para los clientes y usuarios, el producto de software a incorporar o modificar es una herramienta para mejorar o hacer posible su actividad, mientras que para los desarrolladores es el objeto central de su actividad.

Casi sin excepción a lo largo del proceso de desarrollo de software se producen alteraciones en los requisitos del mismo (ver sub-sección 1.1.4) que son el aspecto de la interacción cliente-proveedor que constituye la mayor fuente de discrepancias entre ambos grupos humanos.

Los desarrolladores perciben los cambios en los requisitos como una suerte de estafa (afectiva o intelectual) o al menos como una falta de ética de los clientes que menosprecian su trabajo alterando las condiciones del mismo, haciendo inutilizable documentaciones, diseños y programas terminados o parcialmente desarrollados.

Los clientes perciben los cambios en los requisitos como algo natural a su actividad, ya que una nueva realidad o una mejor percepción de la misma, aunque ella no haya efectivamente cambiado, hacen necesario los cambios que demandan.

En caso de existir alguna resistencia al cambio por parte de los desarrolladores, ya sea por la existencia de una negativa en el incremento del plazo de entrega y/o en el incremento de los costos contractuales, los clientes perciben este hecho como una rigidez que los condena a trabajar de una manera que no es la que pensaban o que ya no les es útil.

Ambas visiones tienen parte de razón y ambas visiones contienen parcialmente elementos falsos. Ambas visiones se contraponen parcialmente y eventualmente dan origen a fricciones porque nacen de un excesivo optimismo de ambas partes.

La visión de los desarrolladores suele ser excesivamente optimista porque supone que su comprensión del sistema a desarrollar es completa y correcta, y también porque no atiende debidamente al hecho que esa realidad cambia durante el lapso de desarrollo.

La visión de los clientes suele ser optimista porque ellos suponen que la hipótesis y los modelos mentales que se hacen acerca de cómo el sistema se va a incorporar al proceso del negocio son completos y correctos. También ellos son afectados por su contexto y se ven obligados a adaptarse al mismo.

La madurez de una organización en relación con sus procesos de incorporación de software no queda definida solamente por la madurez del proceso de desarrollo del mismo sino también por la madurez de los clientes y usuarios en la comprensión de las dificultades involucradas en el proceso.

Es responsabilidad de ambos grupos humanos entender por qué cambian los requisitos y el impacto que dichos cambios tienen sobre la organización y sobre el proceso de desarrollo. Es entonces responsabilidad de los desarrolladores lidiar con estos cambios considerando desde el principio el hecho que inexorablemente se van a producir. Es responsabilidad de los clientes procurar describirlos lo más tempranamente posible y ponderar apropiadamente su importancia.

2.2.3. Características y Actividades del proceso de la IR

La Ingeniería de Requisitos establece el proceso de definición de requisitos como un proceso en el cual se elicitan, modelan y analizan los requisitos [Leite 94]. Este proceso debe lidiar con distintos puntos de vista y usar una combinación de métodos, herramientas, procedimientos y personal. El producto final de este proceso es un documento denominado "Especificación de los Requisitos de Software" (SRS) (ver sub-sección 2.6.5) que puede estar acompañado de otros documentos y modelos, pero el objetivo primario de este proceso es obtener una comprensión acabada del problema de nuestros clientes para establecer la solución más adecuada al problema. Estos otros documentos complementarios suelen estar dedicados a modelar el contexto en el que el software se va a desempeñar y su gran contribución es agregar semántica al SRS.

Este proceso ocurre en un contexto previamente definido: el UdeD, el cual por su naturaleza evoluciona pero además se verá alterado por el proceso mismo de desarrollo del software y posteriormente por su puesta en operación. Dado lo cual, para que la definición de requisitos sea de la mejor calidad posible, el ingeniero de requisitos debe entender este UdeD.

Desde otro punto de vista, se debe considerar que el proceso de la IR, como todo proceso, transforma entradas en salidas. Las entradas corresponden a información contenida dentro de ese UdeD cambiante, heterogéneo, disperso e imperfecto; esa información entonces posee diversas características contrapuestas: probablemente desorganizada, relevante e irrelevante, con distinta granularidad, correcta e incorrecta, coherente y contradictoria, abierta / transparente y oculta / obscura, directamente entendible y requerir interpretación / traducción, vigente y desactualizada, proveniente de diversas fuentes de información, distintos medios de soporte, entre otros aspectos. Mientras que las salidas deben corresponder a información organizada, consistente, lo más completa posible, correcta, sin ambigüedades, cuyo medio de soporte es habitualmente el documento de definición de requisitos (SRS) y los modelos que describen el ambiente y el sistema desde distintas perspectivas y para distintos destinatarios. De lo dicho se desprende que, la transformación de dichas entradas en estas salidas es una tarea ardua que requiere tiempo, personal adiestrado, técnicas adecuadas y herramientas de apoyo.

El proceso de producción de requisitos, a veces denominado proceso de definición o especificación[8] de requisitos, es naturalmente iterativo e involucra las siguientes actividades: la elicitación[9], el modelado y el análisis de los requisitos, y una cuarta actividad cruzada que es la gestión de requisitos (ver

[8] Algunos autores consideran el término especificación como la definición de los detalles técnicos que deberán atender a los requisitos. Esta tesis considera especificación como la representación de un requisito, ver sub-sección 2.6.1.

[9] Elicitar: descubrir, tornar explícito, obtener el máximo de información para el conocimiento del objeto en cuestión.

Figura 2-1, adaptada de [Sábat Neto 00]). Estas actividades interactúan entre sí desarrollándose concurrentemente, por ejemplo, a medida que se va adquiriendo conocimiento, se va modelando y analizando la información recolectada. La Figura 2-2 (actualizada a partir de [Leite 94]) muestra la iteración entre estas actividades por retroalimentación junto con la evolución del UdeD, mientras que la gestión de requisitos también involucra a estas tres actividades a lo largo del tiempo. Es decir, el proceso de definición de requisitos continúa durante todo el ciclo de vida del software.

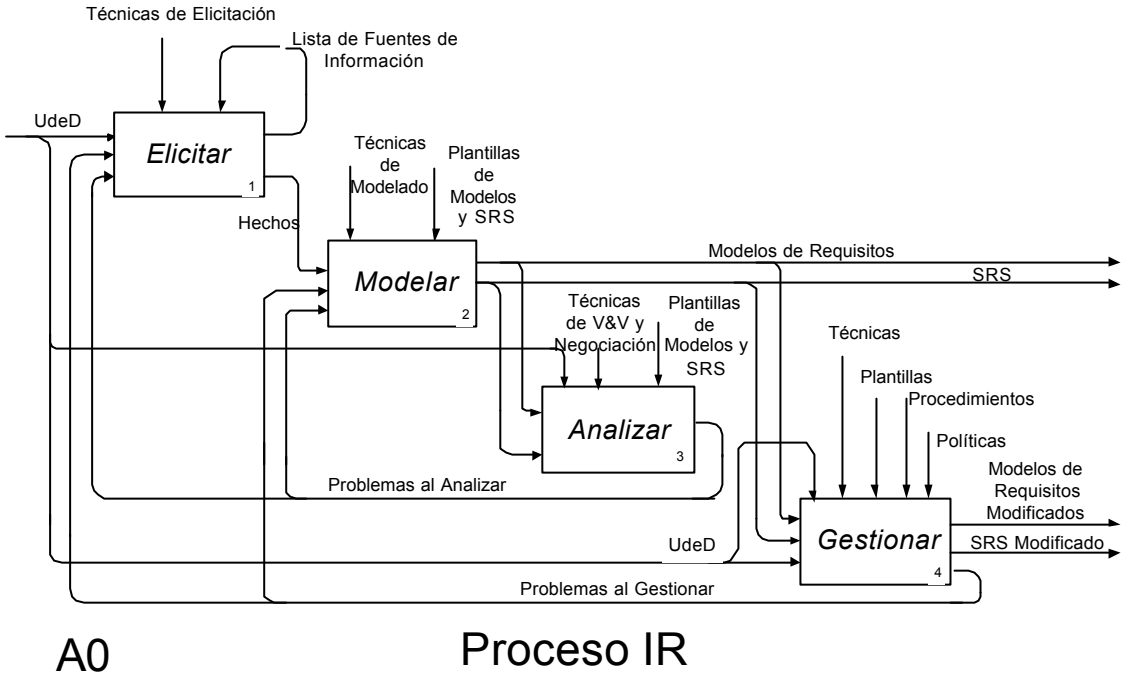


Figura 2-1. Proceso de la Ingeniería de Requisitos

Para un modelo de cascada, es muy difícil determinar cuándo ha finalizado el proceso de definición de requisitos y puede comenzar la etapa de diseño [Regnell 99]. Es una labor muy subjetiva establecer el criterio que determina si el documento de requisitos incluye lo necesario y suficiente para continuar con la siguiente etapa de desarrollo, aún cuando existen investigaciones sobre la evaluación y uso de reglas de finalización ("stopping rules") en la elicitación de requisitos [Pitts 99]. Por otro lado, para un modelo iterativo e incremental, siempre existe la posibilidad de mejorar la definición de los requisitos en sucesivos ciclos de iteración (ver sección 2.3: Modelos de Procesos).

En el nivel 3-definido de CMMI [CMMI 06], donde los procesos deben estar bien caracterizados y descriptos en métodos, procedimientos y herramientas, se presenta el área Desarrollo de Requisitos cuyo propósito es producir y analizar requisitos relativos al cliente, al producto y a los componentes de software. El Desarrollo de Requisitos incluye las siguientes actividades:

i) Elicitación, análisis, validación y comunicación de las necesidades de los clientes, expectativas y restricciones para obtener los requisitos

de clientes.

ii) Reunir y coordinar las necesidades de los involucrados.

iii) Desarrollo del ciclo de vida de los requisitos del producto.

iv) Establecer los requisitos de clientes.

v) Establecer los requisitos del producto y asociados a componentes de software, consistentes con los requisitos de clientes.

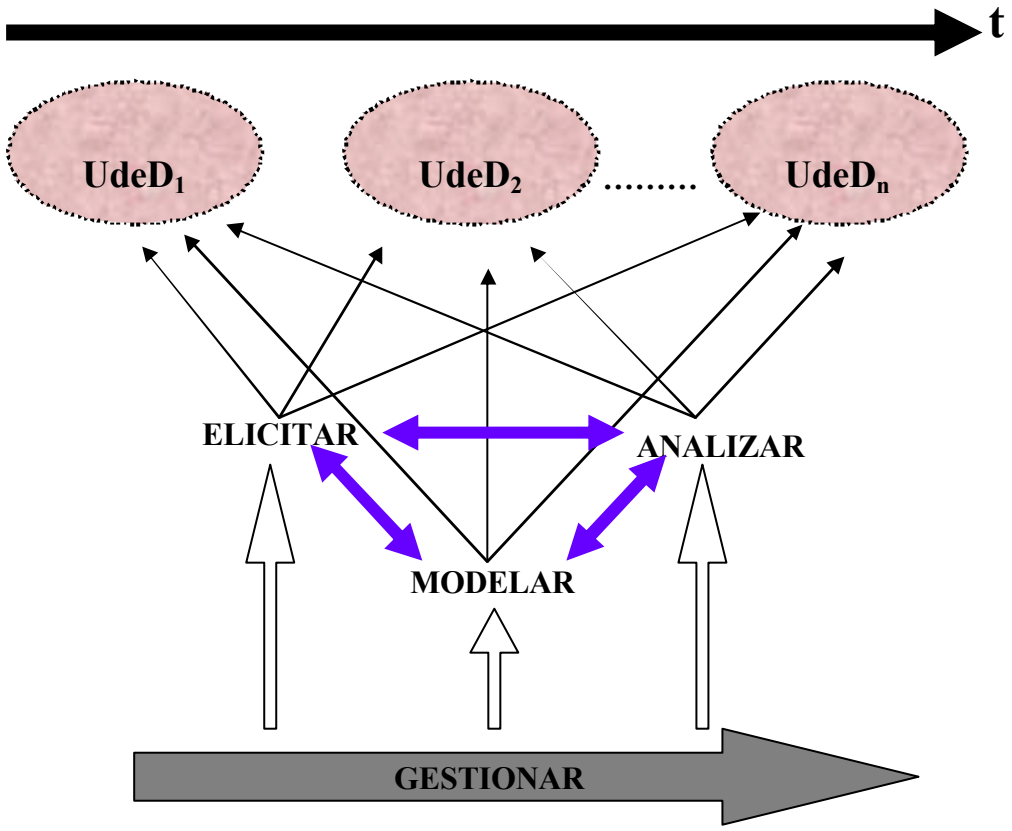


Figura 2-2. Evolución del UdeD frente a las Actividades de la IR

A continuación se describen sumariamente las actividades del proceso de la IR, las cuales se detallan en el contexto de esta tesis en las cuatro sub-secciones siguientes.

Elicitación: se encarga de encontrar los hechos relevantes de la aplicación. Es decir, es una fase de adquisición de conocimiento. Primeramente, se identifican y priorizan las fuentes de información, se seleccionan las técnicas de elicitación más adecuadas al problema en cuestión y luego, comienza la recolección de hechos propiamente dicha. En esta fase es crucial la comunicación que se establece entre los involucrados.

Modelado: se encarga de representar, organizar y registrar (almacenar) los hechos recolectados durante la elicitación. Es decir, éstos deben documentarse en una forma organizada, comprensible y significativa. El modelo puede estar compuesto de distintos conjuntos de representaciones. Se

han adaptado muchos modelos del diseño para modelar requisitos, mientras que se han creado modelos a partir de las necesidades propias de la fase de definición de requisitos.

Análisis: debe verificar y validar los hechos modelados, y acordar los requisitos. Se encarga de descubrir y resolver los problemas que se presentan con los requisitos, por ejemplo: requisitos omitidos, contradictorios, no posibles, superpuestos, ambiguos, erróneos, aplicando diversas técnicas de verificación y validación. Asimismo, esta actividad involucra la negociación de requisitos entre los involucrados. La necesidad de negociar o acordar requisitos puede tener muy diversos orígenes, siendo los más frecuentes la elaboración de propuestas alternativas de los ingenieros de requisitos o la presencia de conflictos preexistentes en las fuentes de información. La negociación involucra entonces el intercambio de información, la discusión, la resolución de conflictos, el acuerdo de propuestas con los clientes y usuarios, y la asignación de prioridades a los requisitos.

Gestión de Requisitos: debe identificar, analizar y realizar los cambios necesarios provenientes de nuevos requisitos o modificaciones a requisitos existentes. Esta actividad se encarga no sólo de identificar nuevos requisitos o cambios en requisitos, sino también identificar los requisitos afectados por estos cambios y estimar los costos del cambio. Una vez efectivizados los cambios en los documentos y modelos de requisitos, estos cambios a su vez deben verificarse y validarse. Es decir, en esta actividad se realizan las tres actividades previamente mencionadas.

2.2.4. Elicitación

Esta actividad consta de tes partes [Leite 94]: la identificación de las fuentes de información, la recolección de hechos y la comunicación. Muchas de las técnicas involucradas en ellas provienen de las Ciencias Sociales [Goguen 93] [Bertolin 98] y por lo tanto, requieren que los ingenieros de requisitos tengan conocimientos de Sociología y Psicología. Las dificultades principales de esta actividad se refieren a la comunicación: uso de lenguajes diferentes provenientes de culturas diferentes, creencias del conocimiento tácito, dispersión del conocimiento, resistencia del usuario a brindar su conocimiento y nivel de habilidad del usuario para transmitir su conocimiento. Es aquí donde los ingenieros de requisitos deben valerse de un conjunto de técnicas de recolección de datos y de comunicación que les permita capturar la mayor cantidad posible de información confiable en función del perfil de los usuarios.

La primera tarea consiste en identificar las fuentes de información, las cuales están contenidas en el UdeD. Por lo tanto, el primer paso es la definición del contexto donde al proceso de IR tendrá lugar. Los límites del UdeD rara vez son precisos, siendo a menudo redefinidos a lo largo del proceso de desarrollo. Una fuente de información obligatoria es el primer documento de objetivos y alcances del sistema (si fue escrito), o los demandantes del sistema de software (los clientes).

Las fuentes de información más confiables son los documentos y las personas involucradas en el UdeD pero otras fuentes importantes pueden ser: libros acerca de temas relacionados, otros sistemas de software del cliente y otros sistemas disponibles en el mercado (COTS). Pero, los usuarios son casi siempre la fuente de información más significativa. El organigrama de la empresa también es un documento útil para identificar personas o departamentos como fuentes de información. Básicamente las fuentes de información [Leite 94] pueden ser:

- ⇨ Personas: clientes, usuarios finales, gerentes, expertos del dominio, consultores.
- ⇨ Documentos del macrosistema (documento preliminar de objetivos y alcance del sistema, organigrama de la empresa, políticas de organización, manual de equipamiento de hardware y/o software, manuales de usuario, manuales de métodos, contratos, formularios, listados, actas de reuniones, entre otros)
- ⇨ Documentos externos al macrosistema (manuales de otros software, libros sobre temas relacionados)
- ⇨ Software interno (sistemas del cliente)
- ⇨ Software externo (otros sistemas ya existentes en empresas o en el mercado)
- ⇨ grupos sociales (formales / informales)

Después de identificar las fuentes de información, se les debe asignar prioridades, principalmente cuando se dispone de una gran cantidad de fuentes de información, pues es casi inviable el acceso a todas ellas. Heurísticas generales para identificar y priorizar fuentes de información [Leite 94]:

i) Identificar los demandantes del software (los clientes).
ii) Identificar la gente que se presume será impactada por la operación del software (los usuarios).
iii) A través de estas personas, identificar otras personas que tengan alguna relación con el macrosistema.
iv) Identificar grupos de interés relacionados con el problema en estudio.
v) Identificar sistemas de software de la organización relacionados con el problema en estudio.
vi) Identificar la existencia de paquetes de software en el mercado.
vii) Identificar los documentos más referenciados por los clientes y usuarios.
viii) Identificar libros u otros documentos u otros sistemas relacionados con el UdeD o mencionados por las personas identificadas.

Burstin [Burstin 84] propuso la construcción de un "árbol de usuarios abstractos" para caracterizar los usuarios y sus necesidades, luego en este árbol, usuarios con necesidades más concretas aparecen en las hojas del árbol. Esto ayuda a identificar el tipo de información a elicitar de los usuarios y permite darles prioridades. También, Zalorenci & Burnett [Zanlorenci 98] muestran un modelo[10] donde los usuarios son calificados como fuentes de

[10] El propósito de calificar usuarios en [Zanlorenci 98] es para evaluar el grado de riesgo de implementar requisitos basados en información elicitada de los usuarios. De todas formas, los tres aspectos para

información teniendo en cuenta tres aspectos: el punto de vista del usuario (consumidor, productor u observador neutral de la información), el nivel de posición en la organización (operativo, gerencial o estratégico) y el grado de demanda de información (esencial, deseable o excedente). Alexander [Alexander 05] propone para identificar involucrados el uso de un "modelo cebolla" de relaciones entre los involucrados, donde cada capa del modelo representa una zona distinta de involucrados. En cada capa se definen clases de roles (por ejemplo Operador Normal) y roles (clases de involucrados, por ejemplo Configurador e Instalador). En Alexander & Robertson [Alexander 04b] el modelo cebolla se complementa con una plantilla de análisis de involucrados, donde se registra cada nombre y sus responsabilidades. En [Leite 07] se propone una estrategia completa para identificar fuentes de información y sus interrelaciones (grafo de relevancia), darles un ranking por relevancia, prioridad y costo (matriz de selección), y asignarles técnicas de elicitación, teniendo en cuenta el punto de vista de cada ingeniero de requisitos.

Técnica	Elicitación	Modelado	Análisis	
			Validación	Negociación
Entrevistas	●			
Cuestionarios	●			
Análisis Documentos	●			
Introspección	●			
Reuniones	●		●	●
Grupos Focalizados	●			●
Prototipación	●		●	●
Escenarios	●	●	●	●
Análisis de Objetivos	●	●		
Análisis de Protocolos	●			
Card Sorting	●			
Observación	●			
Etnografía	●			
Etnometodología	●			
Análisis conversación	●			
Análisis de Dominios	●		●	
Reuso de Requisitos	●			
Ingeniería Reversa	●	●		

Tabla 2-1. Uso de Técnicas en el proceso de la Ingeniería de Requisitos

En función de las fuentes de información identificadas, se debe seleccionar la combinación de técnicas de recolección de hechos [Goguen 93] [Loucopoulos 95] [Kotonya 98] [Whitten 03] más adecuada. Cabe notar que las técnicas de recolección están fuertemente influenciadas por la fuente de información. Las técnicas tradicionales, aquellas más utilizadas, son entrevistas estructuradas y desestructuradas, cuestionarios, análisis de documentación e

calificar usuarios pueden aplicarse casi perfectamente para darles prioridades y seleccionarlos como fuentes de información.

introspección. Dentro de la categoría de técnicas grupales o colaborativas están las reuniones, sesiones de brainstorming y grupos focalizados, como por ejemplo: JRP[11] (Joint Requirements Planning) que son sesiones para acelerar el proceso de definición de requisitos. Otra categoría son las técnicas de prototipación basadas en la construcción de un artefacto ejecutable. Existen técnicas desarrolladas más recientemente que involucran un mayor entrenamiento por parte de los ingenieros de requisitos y corresponden a las tres categorías siguientes. Las técnicas dirigidas por modelos son por ejemplo escenarios y análisis de objetivos. Las técnicas cognitivas son entre otras el análisis de protocolos y la clasificación de tarjetas. Las técnicas contextuales incluyen la observación de participantes, la etnografía, la etnometodología y el análisis de conversación. Estas seis categorías de técnicas de elicitación fueron propuestas por Nuseibeh y Easterbrook en [Nuseibeh 00], sin embargo Loucopoulos en [Loucopoulos 95] propone una categoría adicional basada en el reuso de requisitos[12] que involucra las siguientes técnicas: análisis de dominios, reuso de especificaciones de requisitos e ingeniería reversa.

La Tabla 2-1 muestra como algunas técnicas de elicitación también son utilizadas como técnicas de modelado, de validación o para la negociación. Tal es el caso de Escenarios que es utilizada en las tres actividades de la IR. Un caso más particular es el de Prototipación que es usada además como técnica de validación, y por otro lado, se presenta como un modelo de proceso de software (ver sección 2.3).

No basta con poseer entrenamiento en las técnicas recién mencionadas, los ingenieros de requisitos deben poder comunicarse adecuadamente con los usuarios, lograr la participación activa de ellos, establecer una retroalimentación en la comunicación, identificar el punto de vista del interlocutor y elaborar el nivel de abstracción adecuado a la información capturada.

2.2.5. Modelado

Como parte de esta actividad se selecciona el o los modelos que permitan representar adecuadamente los hechos recolectados. Existen diversas formas de modelar los requisitos [Loucopoulos 95] [Thayer 97] [Kotonya 98] [Sommerville 02] a través de modelos conceptuales[13]: modelos de datos semánticos, modelos del análisis estructurado, lenguajes de especificación formal, modelos organizacionales, modelos orientados a objetos y modelos basados en lenguaje natural, entre otros. Esta agrupación de modelos no es disjunta, sólo sirve para el propósito aclaratorio de este punto.

[11] Estas sesiones de trabajo a veces son denominadas en la literatura como sesiones JAD, aunque éstas últimas cubren un espectro más amplio en sus objetivos (ver sección 2.4).

[12] Esta categoría podría también incluirse dentro de la categoría de técnicas tradicionales, aunque no están dentro de las más utilizadas en la práctica.

[13] Un modelo conceptual da una descripción de los posibles estados de las cuestiones del UdeD, incluyendo clasificaciones, reglas, leyes, etc. del UdeD [Loucopoulos 95]. Utiliza términos semánticos y mecanismos de abstracción para modelar una aplicación. Por lo tanto, los modelos conceptuales son más expresivos y dan una representación más directa y natural de la aplicación que los modelos físicos y modelos lógicos [Mylopoulos 98].

Ejemplos de modelos de datos semánticos son el modelo de entidad-relación [Chen 76], RM/T [Codd 79] que es una extensión del modelo de Chen que incluye relaciones de generalización, SDM (Semantic Data Model) [Hammer 81] que incluye generalización, agregación y agrupamiento.

Dentro de los modelos del análisis estructurado se puede mencionar: actigrama y datagrama de SADT [Ross 77], DFD (diagrama de flujo de datos) [DeMarco 78] [Gane 79] que muestra gráficamente la transformación de datos entre procesos, almacenamientos y agentes externos, diagrama de descomposición [Gane 79] o gráfico de jerarquías que muestra la estructura funcional descendente de un sistema, SA/RT (Structured Analysis / Real-Time) [Ward 85] diagrama de control de flujo para sistemas empotrados de tiempo real que además de involucrar los mismos elementos de un DFD agrega notaciones para proceso de control y flujo de control discreto y continuo, diccionario de datos [DeMarco 78] que es un repositorio que describe cada uno de los almacenamientos y flujos de datos que aparecen en el conjunto de DFDs, utilizando una notación algebraica, diagrama de acceso a datos para describir la estructura y contenido de almacenamiento de datos [Gane 79].

Ejemplos de modelos organizacionales son: KAOS (Knowledge Acquisition in Automated Specification of Software) [Dardenne 93] que modela objetivos, agentes, alternativas, eventos y acciones realizadas por agentes en el ambiente, entre otros conceptos; la técnica i* [Yu 93] incluye dos modelos: el modelo de dependencias estratégicas, que describe las relaciones entre actores para satisfacer objetivos y el modelo de justificación estratégica, que describe la razón de las relaciones entre actores; Telos [Mylopoulos 90] se basa en un esquema de clasificación, que ofrece facilidades de meta-modelado permitiendo definir conceptos.

Son lenguajes de especificación formal: CIM (Conceptual Information Model) [Bubenko 80] incluye una ontología de entidades y eventos, y un sub-lenguaje de especificación de restricciones, RML (Requirements Modeling Language) [Greenspan 82] toma ideas de representación del conocimiento y de los modelos de datos semánticos, y organiza las entidades y actividades en jerarquías de generalización, el modelo ERAE [Dubois 86] tiene dos partes: una parte declarativa donde se identifican y clasifican conceptos y una parte de sentencias donde los conceptos se restringen usando un lenguaje de primer orden, Z [Spivey 89] es un lenguaje de especificación de tipos de datos abstractos, Albert II [Dubois 94], CONGOLOG [DeGiacomo 97] para especificar procesos concurrentes, Petri-nets [Petri 73] [Peterson 77] para especificar procesos dinámicos discretos, entre otros.

Dentro de los modelos orientados a objetos[14], existen modelos que describen la estructura de clases / objetos y otros que describen las relaciones dinámicas entre objetos. En el primer grupo están el diagrama de estructura [Coad 91] que identifica relaciones de agregación y generalización / especialización entre clases, el modelo de información del enfoque de Shlaer y

[14] Se debe hacer notar que muchos de estos modelos están muy orientados al diseño.

Mellor [Shlaer 88] que describe los objetos del sistema, sus relaciones y atributos, e incluye objetos asociativos, el gráfico de jerarquías [Wirfs-Brock 90] que es un modelo de clasificación que muestra relaciones de herencia, la tarjeta de especificación de clase de la técnica CRC (Class-Responsability-Collaboration) [Wirfs-Brock 95] que lista los servicios (responsabilidades) de una clase y los servicios que usa de otras clases (colaboradores), y el diagrama de clases de OMT [Rumbaugh 91]. Ejemplos de modelos dinámicos son: el diagrama de servicios [Coad 91] que describe cada servicio que provee un objeto e identifica mensajes entre objetos, el diagrama de transición de estados [Coad 91] [Shlaer 88] [Rumbaugh 91] [Booch 92], el modelo de ciclo de vida del método Fusion [Coleman 94] que usa expresiones regulares para describir secuencias de interacciones entre el sistema y el ambiente, el modelo de interacción entre objetos [Shlaer 88] [Wirfs-Brock 90] [Booch 92], y diagrama de casos de uso del método OOSE (Object Oriented Software Engineering) de Jacobson [Jacobson 92]. Rational Unified Process [Jacobson 99] [Kruchten 04] incluye muchos de los modelos utilizados en los métodos orientados a objetos mencionados arriba, agregando algunos otros modelos como: diagrama de actividades, diagrama de paquetes y diagrama de plataforma, pero describiendo todos sus modelos utilizando la notación UML [Booch 99] [Rumbaugh 05], que fue creada originalmente por Rational Software para comunicar requisitos, arquitecturas y diseños, y ahora es mantenida por la organización de estándares OMG (Object Management Group) [UML 03].

Los modelos de lenguaje natural son de uso mucho más reciente y se han incorporado en varios métodos de la IR. Están entre otros: los Casos de Uso [Jacobson 92] descripciones narrativas de la interacción entre un actor y el sistema, los Escenarios [Booch 92] [Potts 94] [Wirfs-Brock 95] descripciones de situaciones del UdeD en diversos formatos según los autores, LEL [Leite 93] glosario con la denotación y connotación de los términos utilizados en el UdeD, glosarios con sólo definiciones de términos del UdeD [Whitenack 94] [Oberg 98] [Kovitz 98], glosarios que describen la terminología utilizada en otros modelos de IR [Rolland 98c] [Regnell 99b] [Alspaugh 99], modelo de Reglas de Negocio [Ross 97] [Leite 98] [Gottesdiener 99], User-Stories [Beck 00] versión simplificada de Casos de Uso para Programación eXtrema, entre otros.

Estos modelos tienen su propia representación y organización que el equipo de ingenieros de requisitos debe conocer y respetar. También debe tenerse en cuenta el almacenamiento de estos modelos para facilitar el registro y recuperación de los hechos elicitados y representados.

Pero los modelos presentan varios problemas, principalmente: las barreras de comunicación, la incompletitud y la inconsistencia. El primer problema está relacionado con la dificultad que se les presenta a los usuarios para interpretar modelos escritos en un lenguaje técnico. Existen distintas técnicas para salvar este problema, por ejemplo la traducción del modelo a un esquema entendible por los usuarios (técnica de parafraseo al lenguaje natural [Loucopoulos 95]), o utilizar modelos directamente escritos en lenguaje natural, tales como los escenarios [Leite 97], casos de uso [Jacobson 92] y el léxico extendido del lenguaje [Leite 93]. El segundo problema, la incompletitud, es un atributo inherente a cualquier modelo, la forma de lograr un modelado lo más

completo posible es mediante el uso combinado de modelos, cada uno con una perspectiva diferente del UdeD, y contar con procedimientos y heurísticas que guíen su construcción. El tercer inconveniente, la inconsistencia, se elimina en gran medida aplicando técnicas de verificación y validación (ver siguiente sub-sección).

2.2.6. Análisis

El análisis de requisitos involucra las siguientes sub-actividades: la verificación de los modelos, su validación con los clientes y usuarios, y la negociación de los requisitos. El objetivo de esta actividad es establecer un conjunto de requisitos consistentes, lo más completos posibles y acordados entre todos los involucrados. Al igual que la actividad de elicitación no sólo requiere por parte de los ingenieros de requisitos de conocimientos en la Ciencia de la Computación sino también de las Ciencias Sociales, pues se requiere una gran participación de los clientes y usuarios, los cuales deben interpretar los modelos y documentos presentados para confirmar su corrección y se deben acordar los requisitos del sistema de software entre todos los involucrados. Es decir, se debe lograr una comunicación fluida con los clientes y usuarios, y además se debe intentar que ellos se sientan compenetrados con el proyecto, ya que el éxito del mismo es debido al esfuerzo cooperativo de todos los involucrados.

Para todas las actividades de análisis, hay una sub-actividad inicial que es la identificación de partes [Leite 94]. Esta actividad es una suerte de planeamiento donde se determina el objeto de análisis, dado que en sistemas complejos se dificulta realizar un análisis de todo el sistema, y se determinan los interlocutores (o participantes) para la validación y la negociación. Una vez identificadas las partes se pueden establecer políticas de análisis respecto a la prioridad de las partes a analizar y la técnica de análisis a aplicar.

Las actividades relacionadas con calidad dentro del proceso de IR son básicamente la verificación y la validación de requisitos. Parafraseando a Boehm [Boehm 84], la V&V de requisitos podría definirse informalmente por las preguntas:

> **Verificación**: «*Am I building the requirements **right***? »
> **Validación**: «*Am I building the **right** requirements*? »

La Figura 2-3 muestra cómo la verificación se realiza entre modelos o dentro del propio modelo, mientras que la validación requiere la confrontación con el UdeD. En otras palabras, la verificación permite determinar si el modelo ha sido construido correctamente y la validación permite determinar si el modelo refleja la realidad.

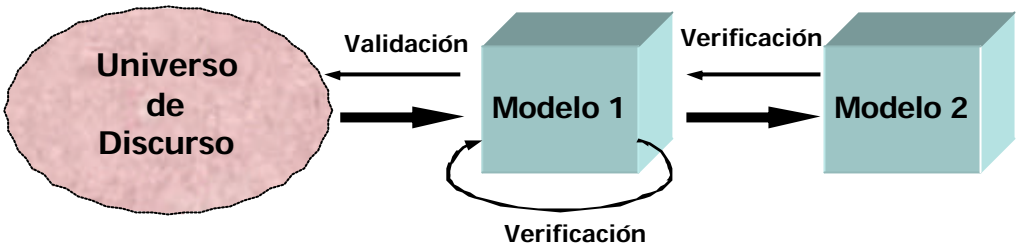


Figura 2-3. Verificación y Validación

Loucopoulos & Karakostas en [Loucopoulos 95] definen a la validación de requisitos como el proceso de certificación de la exactitud del modelo de requisitos comparado con la intención de los usuarios y enfatiza que en contraste con la verificación, la validación no puede ejecutarla el analista con herramientas de software solamente; la activa participación de los usuarios es siempre necesaria.

Para Sommerville & Sawyer en [Sommerville 97] la validación de requisitos debe definirse como el chequeo de que el documento de requisitos final incluya todos[15] los requisitos del sistema y no posea aspectos incompletos ni inconsistentes. El proceso de validación sólo puede incrementar la confianza de que las especificaciones, que representan al sistema, van a representar las necesidades reales del usuario del sistema.

Sin embargo, Kotonya & Sommerville en [Kotonya 98, pág. 88] definen en forma totalmente opuesta a la verificación (denominada "análisis") de la validación. Presentan en forma inversa las dos preguntas expuestas arriba de Boehm, para distinguir la verificación de la validación. Existen en la literatura algunos otros casos contradictorios sobre V&V o donde directamente no se realiza esta distinción (y en general emplean el término validación). Esta tesis se adhiere a la definición presentada en [Loucopoulos 95].

La mayoría de estas técnicas son adaptaciones de técnicas provenientes de las fases de diseño, de codificación y/o de prueba [Laitenberger 00].

Observando distintos autores [Loucopoulos 95] [Wallace 97] [Basili 81] que presentan técnicas de V&V para modelos y documentos de requisitos, y el estándar IEEE sobre Especificación de Requisitos de Software [IEEE Std 830-1998], se desprende una variedad disímil de propiedades que deben examinarse durante una verificación o validación de requisitos. Davis et al. [Davis 93b] generó una lista de 24 propiedades que debe presentar un SRS de calidad: no ambiguo, completo, correcto, entendible, verificable, consistente internamente, consistente externamente, alcanzable (realizable), conciso, independiente de diseño, rastreable, modificable, almacenable electrónicamente, ejecutable / interpretable, registrable por importancia relativa, registrable por estabilidad relativa, registrable por versión, no redundante, en el nivel de detalle adecuado, preciso, reusable, remontado a su origen,

[15] Ver Nota al Pie previa: Falacia de la Completitud. Tener presente a lo largo de la tesis esta observación cada vez que se menciona el tema de modelos o documentos completos o con todos sus elementos.

organizado y referenciado cruzadamente. Es obvio que cada técnica en particular ya sea de verificación o validación atiende un conjunto discreto de estas propiedades.

La Tabla 2-2 enumera las principales técnicas de verificación y validación de modelos y documentos de requisitos. Cabe aclarar que muchas veces en la literatura cuando se habla de verificar y principalmente validar requisitos, se está haciendo referencia a chequear el sistema de software contra los requisitos, ya sean éstos soportados en un documento formal o directamente contra el UdeD. Mientras que la actividad a la que se hace referencia dentro de la IR atiende a verificar y validar el documento o modelos que soportan los requisitos, previo al diseño y codificación del sistema o componente de software. Es decir, esta actividad comienza bien temprano en el ciclo de vida del software. La Tabla 2-2 fue elaborada a partir de [Wallace 97] adaptada según las premisas recién mencionadas e incorporando otras técnicas de V&V extraídas de la literatura [Loucopoulos 95] [Kotonya 98] [Leite 94] [Sommerville 97].

Técnica	Verificación de requisitos	Validación de requisitos
Animación		●
Análisis de Algoritmos	●	
Análisis de Límites		●
Chequeo contra Estándares	●	
Análisis de Checklists	●	
Análisis de Consistencia	●	
Análisis de Flujo de Control	●	
Reuso de Dominios	●	
Enfoque basado en Sistemas Expertos		●
Máquinas de Estado Finitas	●	
Definición de Checklist		●
Inspecciones	●	
Matrices de Interacción	●	
Parafraseo de Lenguaje Natural		●
Generar Casos de Prueba para requisitos		●
Uso de Prototipos		●
Análisis de Puntos de Vista		●
Parsing de Requisitos	●	
Revisiones	●	●
Walkthroughs	●	
Escribir borrador del Manual de Usuario		●

Tabla 2-2. Técnicas de V&V en la Ingeniería de Requisitos

En general, algunas técnicas de verificación pueden ser automatizadas dado que no requieren la intervención de los clientes y usuarios, salvo en los casos de verificaciones semánticas donde puede requerirse la intervención humana, pero de los ingenieros de requisitos. Por el contrario, la validación

requiere la participación activa de los clientes y usuarios, para establecer la correspondencia entre el modelo y la realidad.

Con respecto a la negociación, ésta es una actividad cuyo objetivo es lograr un acuerdo de los requisitos del software entre todos los involucrados. Es una actividad que se solapa con la elicitación y con la validación, pero que conceptualmente puede diferenciarse claramente de ellas pues busca un compromiso en el cual todas las partes involucradas concuerden. La negociación abarca dos aspectos de naturaleza bien distinguible: i) evaluar y decidir sobre las propuestas que elaboran los ingenieros de requisitos, frecuentemente denominadas en la literatura, alternativas o soluciones candidatas[16], y ii) estudiar y resolver diferencias de intereses entre los involucrados, las cuales promueven requisitos contradictorios o no factibles.

Es inevitable que surjan conflictos en los requisitos cuando en un proyecto de software participan muchos clientes y usuarios, que frecuentemente tienen diferentes intereses ya sea personales o por su rol en la organización. Asimismo el grupo de ingenieros tiene otros tipos de intereses en el proyecto, los cuales pueden chocar o divergir respecto a los de los clientes y usuarios, pues ellos tienen presiones en cuanto a tiempos, costos y recursos, como así también sus propias preferencias y experiencias personales. Los ingenieros deben cumplir con los objetivos del software y con los objetivos del proyecto de desarrollo, bajo restricciones al proyecto y al propio software; mientras que en general los usuarios tienen demandas ilimitadas, sólo restringidas por su imaginación o experiencia. Es bajo estas circunstancias que la negociación debe llevarse a cabo, siendo tarea de los ingenieros llevar esas demandas a la realidad del dominio de la aplicación.

Pueden ocurrir contradicciones entre las necesidades de los usuarios, o demandas que entren en conflicto con los objetivos de la organización, o puede haber requisitos muy ambiciosos y no todos pueden satisfacerse de acuerdo al presupuesto de un proyecto, tanto en tiempo como costos. Se negocia entonces la criticidad y la prioridad de los requisitos. Todos estos conflictos, aunque de distinta naturaleza, no son defectos pero reflejan necesidades y prioridades diferentes entre las partes interesadas. Es frecuente la detección de conflictos entre requisitos no funcionales (ver sub-sección 2.6.4), como por ejemplo entre requisitos de mantenibilidad y de eficiencia, o entre requisitos de seguridad y de disponibilidad [Kotonya 98, p.239].

La negociación no sólo ocurre por conflictos en los requisitos sino también para evaluar las propuestas de los ingenieros de requisitos frente a determinadas circunstancias. Estas propuestas pueden o no estar en conflicto con los requerimientos de los usuarios. Es importante destacar que no necesariamente una propuesta del ingeniero de requisitos está asociada a un interés personal o del grupo de desarrollo de facilitar su tarea, sino que existen muchas y muy sutiles causas que dan origen a propuestas, como atacar problemas o incluir mejoras según la visión que tienen los ingenieros de requisitos respecto al futuro sistema. En forma introductoria puede decirse que

[16] Estos términos hacen referencia a propuestas a nivel del negocio, prescindiendo de formas alternativas de diseñar soluciones basadas en tecnología.

las principales fuentes de propuestas son la intención de:

i) simplificar el software y/o el proceso del negocio,
ii) mejorar el proceso del negocio,
iii) precisar el proceso del negocio y, como consecuencia, el software.

En los dos primeros casos ocurre el fenómeno que la percepción diferenciada del software y del proceso del negocio de los clientes y usuarios en relación con los ingenieros de requisitos hace que vean una mínima peculiaridad del negocio con dos ópticas distintas. Es así, que los clientes y usuarios demandan un servicio del sistema futuro cuya necesidad no es completamente compartida por los ingenieros de requisitos. Idealmente, si los ingenieros de requisitos pueden comprender en profundidad el objetivo subyacente, es posible que puedan proponer una forma de trabajo diferente que satisfaga ese mismo objetivo y que sea, según su propia visión, más coherente o consistente con la concepción global del sistema que ellos perciben. En alguna medida, el primer y segundo caso se pueden diferenciar por el interés en la propuesta, ya que en el primer caso existe un propósito explícito de simplificar el desarrollo posterior del software.

En el tercer caso, las propuestas tienen origen en la necesidad de precisar algún aspecto no claro en el proceso del negocio o en la percepción de los propios ingenieros de requisitos. La necesidad de "llenar un hueco" conceptual es muy frecuente cuando el proceso del negocio se modificará substancialmente como consecuencia del sistema de software y precisamente gracias a ese sistema de software. Cuando esto ocurre, se tiene que se ha detectado la existencia de incertidumbres menores o sutiles en algún aspecto del proceso del negocio que no trasunta importantes diferencias de objetivos pero cuya definición se torna indispensable al momento de desarrollar un software que soporte dicho proceso.

Las propuestas de los ingenieros habitualmente no producen variaciones en los objetivos del negocio. Estas propuestas pueden aportar mejoras menores en los procesos del negocio; en general, las mejoras sustanciales e incluso cambios radicales vienen como consecuencia de un estudio previo abocado explícitamente a tal fin, como ser una reingeniería de los procesos del negocio [Hammer 90]. La conjunción de una reingeniería con una incorporación de un sistema de software que soporte los procesos del negocio produce una gran proliferación de propuestas por parte de los ingenieros de requisitos, las cuales deben ser puestas en consideración principalmente a los clientes y a los niveles gerenciales. Obviamente las propuestas que involucran cambios menores en los procesos también deben ser consensuadas, pero generalmente a niveles gerenciales y operativos.

Resumiendo, la negociación involucra, por un lado, el tratamiento de propuestas de requisitos generadas por los propios ingenieros y, por otro lado, la resolución de conflictos entre clientes, usuarios e ingenieros de requisitos debido a objetivos o necesidades contrapuestas y/o recursos limitados. Como se esquematiza en la Figura 2-4, las actividades involucradas en la negociación

son entonces:

i) *Análisis de conflictos*: los conflictos de intereses detectados durante la V&V son evaluados por los distintos interesados junto con los ingenieros de requisitos[17].

ii) *Evaluación de propuestas*: las propuestas concebidas por los ingenieros de requisitos son puestas en consideración y estudiadas respecto a la causa que le dio origen, ya sea ésta un intento de modificar una actividad del proceso del negocio, de precisar un aspecto no totalmente definido en el UdeD o de presentar alternativas sobre uno o varios requisitos.

iii) *Conciliación de requisitos*: su objetivo es establecer un compromiso sobre los requisitos del software entre todos los involucrados, y comprende la resolución de conflictos, la toma de decisión sobre las propuestas (aceptación o rechazo) y el establecimiento de prioridades para los requisitos acordados.

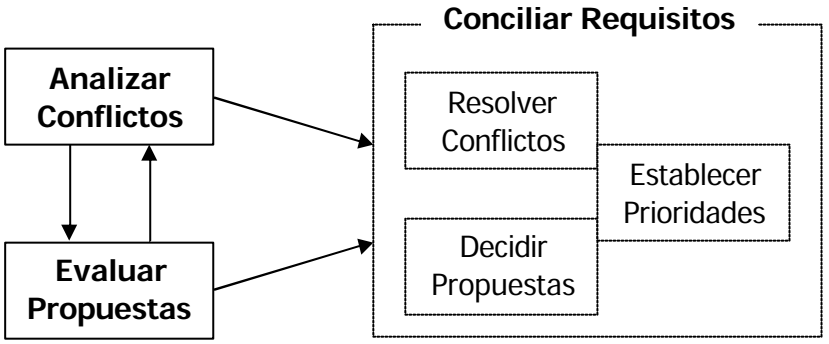


Figura 2-4. Sub-actividades de la Negociación

La técnica ampliamente utilizada para la conciliación es la reunión [Kotonya 98]. Los ingenieros de requisitos encuentran contradicciones entre los requisitos por conflictos de intereses durante la verificación y/o validación de los mismos, y por otro lado, generan propuestas de requisitos durante el modelado. Finalmente llevan a cabo reuniones con los clientes y usuarios que pueden ayudar a resolver dichos conflictos y/o decidir sobre las propuestas. Es conveniente que cada conflicto o propuesta se discuta individualmente. El establecimiento de prioridades a los requisitos puede ser simple o complejo, desde una asignación que califique al requisito como obligatorio o postergable (suspendido) hasta un ranking de importancia o criticidad de los requisitos. Existen variadas técnicas de priorización de requisitos, como por ejemplo: AHP (Analytic Hierarchy Process) [Saaty 80] [Karlsson 98] basada en la técnica de "pairwise comparison" estimando un valor relativo de importancia de los

[17] Existen diversas estrategias para la identificación y análisis de conflictos, como por ejemplo: uso de matrices de interacción [Kotonya 98], Root Requirements Analysis: técnica que identifica requisitos claves y su interacción, propuesto por William Robinson [Robinson 04], Systematic Tradeoff Analysis: técnica basada en lógica difusa propuesta por Yen y Tiao [Yen 97], Viewpoint-Oriented System Engineering (VOSE) [Nuseibeh 94]: método que utiliza templates de puntos de vista, Viwepoint Resolution [Leite 91b]: compara diferentes vistas y provee un esquema para negociar conflictos, entre otras estrategias.

requisitos, QFD (Quality Function Deployment) [Zultner 92] basada en la asignación numérica de prioridades respecto a una escala absoluta, y EVOLVE [Greer 05] basada en un algoritmo genético que considera distintos puntos de vista de los involucrados, restricciones de esfuerzo, restricciones de riesgo y dependencias entre requisitos.

2.2.7. Gestión de Requisitos

Es la actividad que administra los cambios en los requisitos. Estos cambios son de dos tipos: cambios en requisitos existentes o aparición de nuevos requisitos. Los factores que provocan estos cambios se analizaron en la sub-sección 1.1.4 (Evolución de los requisitos), pero básicamente son la evolución del UdeD, incluyendo cambios en las expectativas de los clientes y usuarios, y el descubrimiento de defectos en los requisitos en subsiguientes etapas del proyecto o ya durante la operación del software.

Cabe acá hacer una salvedad sobre el uso del término "Gestión de Requisitos". Existe a veces una confusión entre lo que es "Gestión de los requisitos" y "Gestión por los requisitos" (denominada en la literatura "Gestión de Requisitos") [Sayão 05] [Leite 06].

La Gestión de los Requisitos es el proceso de gestión que se debe utilizar para producir los requisitos. La Gestión por Requisitos (o Gestión de Requisitos) está asociada al proceso de controlar todo el proceso de desarrollo teniendo como referencia el baseline de requisitos. Es decir, la producción de requisitos necesita de un proceso gerencial que permita que esa tarea sea bien realizada. Mientras que la gerencia con base en requisitos es una actividad perenne a lo largo del proceso de producción. La confusión puede residir en que el proceso de Gestión por Requisitos impacta en el propio proceso de Gestión de los Requisitos, dado que la Gestión por Requisitos provee una importante retro-alimentación para la Gestión de los Requisitos. Las actividades del proceso de Gestión de los Requisitos se encuentran en el nivel 3-Definido de CMMI [CMMI 06] en el área Desarrollo de Requisitos (ver sub-sección 2.2.3), mientras que las actividades del proceso Gestión por los Requisitos se encuentran en el nivel 2-Repetible de CMMI en el área Gestión de Requisitos (ver más adelante en esta misma sub-sección).

Para que la administración de cambios sea una actividad viable y exitosa, es indispensable la administración de las dependencias entre los requisitos y la administración de las vinculaciones entre el documento de requisitos y otros documentos, modelos y componentes del software [Kotonya 98] [Davis 99]. Pues un cambio o incorporación de un único requisito puede alterar un conjunto de requisitos en cascada, y dependiendo de la etapa en que se encuentre el proyecto es necesario la actualización de la documentación y otros artefactos del software, lo cual debe identificarse y evaluarse previo a su efectiva realización. Esto está intrínsecamente relacionado con un concepto clave: la rastreabilidad de los requisitos ("requirements traceability") (ver sub-sección 2.2.8). Por otro lado, esta actividad puede ser considerada desde la visión del ciclo de vida del software como parte de una actividad de mayor

alcance: la Gestión de Configuración[18][Crnkovic 99].

La actividad de administración de los cambios se encarga de controlar, documentar y realizar los cambios en los documentos y modelos de requisitos, e involucra las siguientes sub-actividades [Kotonya 98] (ver Figura 2-5):

i) *Identificación de cambios*: consiste en identificar algún problema en los requisitos o un pedido de modificación o de nuevas necesidades proveniente de los clientes y usuarios, y proponer los cambios en los requisitos. Con frecuencia, se suele llenar un formulario de pedido de cambio, que identifica la fuente solicitante y la descripción del cambio mismo, y se utiliza en las siguientes sub-actividades para registrar lo realizado respecto a cada pedido.

ii) *Análisis y Costeo de cambios*: consiste en analizar la validez de los cambios propuestos, identificar los requisitos afectados por los cambios y, si corresponde, los cambios en los componentes de software, estimar, evaluar y negociar el impacto del cambio en el alcance, el cronograma y el costo, y determinar la aceptación o rechazo del cambio.

iii) *Realización de cambios*: consiste en implementar los cambios en los documentos de requisitos y en los modelos, así como verificar y validar estos cambios.

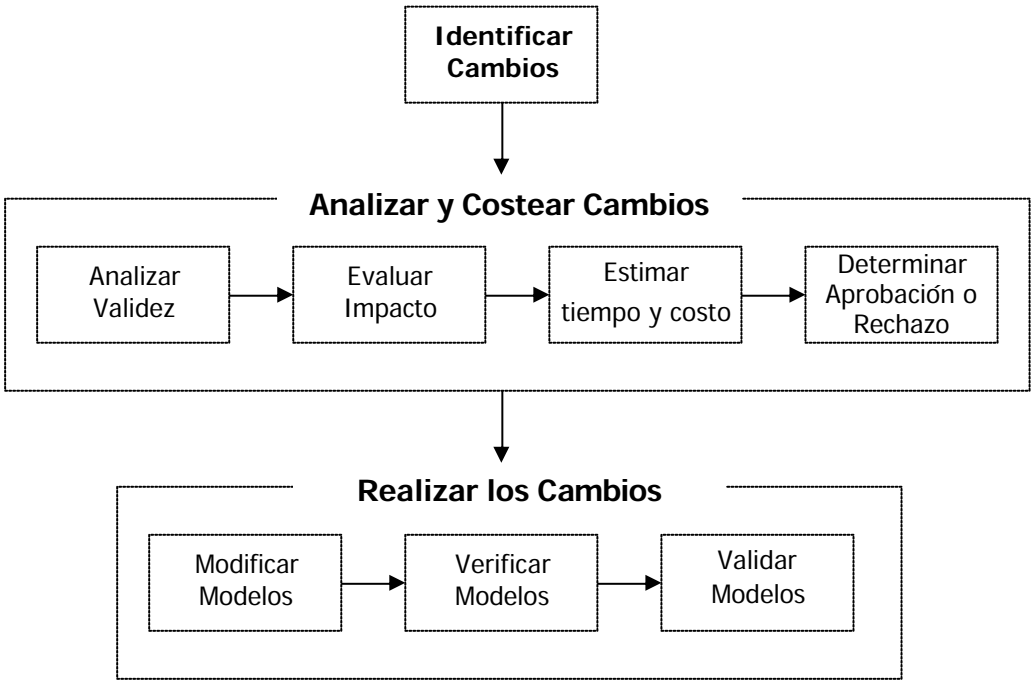


Figura 2-5. Sub-actividades de la Gestión de Requisitos

Para identificar los requisitos afectados por los cambios, así como los documentos, modelos y componentes afectados, se debe utilizar la información de rastreabilidad. Es fundamental contar con alguna herramienta de soporte a

[18] La Gestión de Configuración se encarga de la administración de los cambios en el software, la administración de versiones y liberaciones de componentes de software, y la generación del sistema de software ejecutable para una configuración particular [Sommerville 02].

la gestión pues generalmente se manejan grandes cantidades de información en distintos formatos, y es necesario conservar el rastro de cada cambio solicitado, en evaluación e implementado.

Esta actividad se rige por políticas, procedimientos y procesos que establecen cómo se maneja un cambio en un requisito [Whitten 03] [Kotonya 98]: cómo debe realizarse un pedido de cambio, cómo se evalúa y costea el impacto del cambio, cómo se determina la aprobación o rechazo, cómo se implementa el cambio si es aprobado, qué información debe ser rastreable y cómo, quiénes son los responsables de cada paso en la gestión, entre otros aspectos. Es decir, se requiere una buena planificación de la gestión de requisitos.

La gestión de requisitos es una actividad esencial para el éxito tanto del proyecto de software como del posterior mantenimiento de dicho software. Tal es así que CMM [CMM 95] incluye a esta actividad como una de las seis áreas de proceso claves para que una organización de desarrollo de software pueda pasar del nivel 1-inicial al nivel 2-repetible. CMM establece que: i) se debe determinar el impacto del cambio antes de realizar el cambio, ii) los cambios deben negociarse y comunicarse a los grupos que serán afectados por el cambio, y iii) todo cambio debe rastrearse hasta ser efectivamente cumplido. Siguiendo los mismos lineamientos, CMMI [CMMI 06] también define a la Gestión de Requisitos como una de las seis áreas de proceso de Ingeniería, estableciendo en el nivel 2-repetible, además de las prácticas generales de cumplimiento de objetivos e institucionalización de este proceso, las siguientes prácticas específicas sobre:

i) Desarrollo de una comprensión de los requisitos con los proveedores de los mismos.
ii) Obtención del compromiso de los participantes con los requisitos.
iii) Manejo de todos los cambios en los requisitos mientras evolucionan.
iv) Mantenimiento de la rastreabilidad bidireccional de los requisitos.
v) Identificación de inconsistencias entre el plan del proyecto, los productos de trabajo y los requisitos.

Otra actividad involucrada en la gestión de requisitos es el control del versionado [Sawyer 01] [CMMI 06], es decir, el mantenimiento de la historia de los cambios en los requisitos. El sistema de versionado [Conradi 98] es un soporte de servicios indispensable para garantizar una adecuada rastreabilidad de los requisitos, da una idea de la evolución de los requisitos, registrando todos los cambios realizados con su justificación. El control de versiones puede establecerse a nivel del documento de requisitos (mantener versiones de SRS), a nivel de grupo de requisitos (secciones o capítulos dentro de SRS) y el más complejo a nivel de requisito individual [Kotonya 98]. Se debe identificar a los requisitos como ítems de configuración y manejarlos bajo el mismo régimen de configuración al igual que otros productos del proceso de desarrollo [Sawyer 01] [Crnkovic 99]. El versionado no se circunscribe exclusivamente al documento SRS sino también a mantener versiones de los modelos de requisitos construidos y, a veces, a nivel de elementos individuales del modelo; esta variedad de elementos (denominados unidad de versión o ítem de configuración) hace que los sistemas de control de versionado sean complejos.

Es conveniente contar entonces con herramientas específicas que provean el control de versionado, y que soporten la integración de tecnologías [Brown 92] [Hamilton 91]. Por otro lado, un sistema de control de versiones debe minimizar el costo y la complejidad de crear y desarrollar nuevas versiones para evitar que pierda su utilidad.

2.2.8. Rastreo de los Requisitos

La rastreabilidad de los requisitos es una función de la gestión de requisitos, que se encarga de mantener vínculos entre requisitos dependientes, entre requisitos, diseño y código, y entre requisitos y fuentes de información que los originaron.

Según [IEEE Std 830-1998], un requisito es rastreable si su origen es claro y si facilita su referencia en la documentación de futuros desarrollos o mejoras. Kotonya & Sommerville ofrecen una definición más precisa [Kotonya 98]: un requisito es rastreable si se puede determinar quién lo sugirió, qué requisitos están relacionados con él y cómo se relaciona con otra información tal como: diseño del sistema, implementaciones y documentación del usuario.

Muchos investigadores han estudiado el tema y realizado propuestas sobre la rastreabilidad de los requisitos [Gotel 94] [Wieringa 95] [Palmer 96] [Jarke 98] [Pinheiro 04]. A continuación se extrae la definición dada por Pinheiro & Goguen [Pinheiro 96] por su claridad y amplitud:

> *«La rastreabilidad de los requisitos se refiere a la habilidad de definir, capturar y seguir las pistas dejadas por los requisitos sobre otros elementos del ambiente de desarrollo del software y las pistas dejadas por dichos elementos sobre los requisitos.»*
>
> [Pinheiro 96]

Como indican Pinheiro & Goguen, los elementos del ambiente de desarrollo incluyen tanto aspectos técnicos (por ejemplo: modelos del sistema, código, documentos) como sociales (por ejemplo: personas, políticas, decisiones, objetivos).

Se han propuesto distintos formas de rastreo de requisitos. Tanto Wieringa [Wieringa 95] como [IEEE Std 830-1998], mencionan dos tipos: Backward traceability (habilidad de rastreo de un requisito a sus fuentes) y Forward traceability (habilidad de rastreo de un requisito hacia los componentes de diseño y código). Mientras que Davis en [Davis 93] menciona cuatro tipos pues agrega el rastreo inverso en cada caso, es decir de una fuente a un requisito y de un componente a un requisito; CMMI [CMMI 06] la denomina rastreabilidad bidireccional. Por otro lado, Pinheiro en [Pinheiro 04] introduce otra clasificación: Inter-requirements traceability (habilidad de rastreo entre requisitos dependientes) y Extra-requirements traceability (habilidad de rastreo entre requisitos y otros artefactos). Como se observa, Wieringa y Davis hacen referencia a este último tipo de rastreabilidad. Para realizar un cambio

en un requisito es necesario poder rastrear tanto interna como externamente el requisito, con el fin de evaluar adecuadamente tanto los requisitos dependientes como los componentes que deberán adaptarse y, en ciertos casos, es importante analizar el origen del requisito. En la Figura 2-6 se muestra un esquema que representa las vinculaciones de rastreo mencionadas.

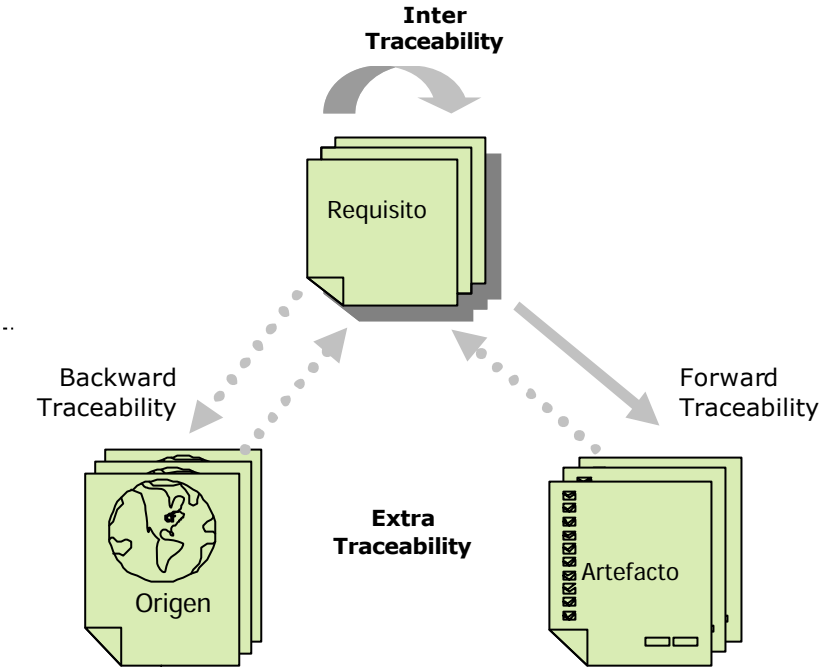


Figura 2-6. Tipos de Rastreabilidad

La recolección y mantenimiento de la información de rastreo es de muy alto costo. Por lo tanto, se deben tener políticas que indiquen qué tipo de rastreos se realizarán y cómo se mantendrá dicha información. Como indica Kotonya & Sommerville [Kotonya 98], la información de rastreo que más habitualmente se mantiene en la práctica es la que corresponde a Inter-requirements traceability y a Forward Traceability desde el documento de requisitos al diseño (marcado con flechas llenas en la Figura 2-6).

La rastreabilidad no sólo se utiliza para administrar los cambios en los requisitos, sino que también es de fundamental ayuda para la verificación y validación de los requisitos y para el control del proceso de desarrollo [Palmer 96] [Davis 99], pues facilita detectar conflictos utilizando los vínculos establecidos entre los elementos rastreables, posibilita asegurar que decisiones tomadas avanzado el desarrollo sean consistentes con decisiones tempranas, y permite verificar que todos los requisitos han sido implementados en el software, entre otros usos. A pesar de los múltiples propósitos que cubre la rastreabilidad de los requisitos, muchas veces sólo es aplicada parcialmente. Esto es debido, por un lado, a su alto costo de producción y mantenimiento (gran diversidad de entidades rastreables) y, por otro lado, a la imperiosa necesidad de contar con herramientas automatizadas que permitan implementar adecuadamente la rastreabilidad.

2.2.9. Comparación de taxonomías de actividades en la IR

No existe un único proceso de la IR, distintas organizaciones adoptan distintos procesos basados en su propia experiencia, en el dominio de la aplicación y en la organización misma; tampoco los organismos que fijan estándares describen un proceso estándar de la IR, sólo describen actividades a cumplir, documentación a generar y prácticas adecuadas.

En el punto anterior, se presentó un proceso de la IR con sus actividades y sub-actividades, pero en la literatura, también se presenta una diversidad de formas de organizar dichas actividades. En general casi todas las sub-actividades están incluidas en el proceso pero bajo distinta agrupación de actividades o simplemente usando distintos nombres. Se describe a continuación estas diferencias tomando como referencia la taxonomía dada por Leite en [Leite 94] y adaptada en esta tesis.

La Tabla 2-3 muestra las actividades en que dividen al proceso de la IR distintos autores: Julio Leite [Leite 94] [Leite 01b], Loucopoulos & Karakostas [Loucopoulos 95], Alan Davis [Davis 96], Kotonya & Sommerville [Kotonya 98], Thayer & Dorfman [Thayer 97b] y la guía propuesta por la IEEE [IEEE Std P1233/D3-1995] para desarrollar una especificación de requisitos.

Leite [Leite 94] [Leite 01b]	**Loucopoulos & Karakostas** [Loucopoulos 95]	**Davis** [Davis 96]	**Kotonya & Sommerville** [Kotonya 98]	**Thayer & Dorfman** [Thayer 97b]	**IEEE P1233/D3**[19] [IEEE Std P1233/D3-1995]
▪ Elicitar ▪ Modelar ▪ Analizar ▪ Gestionar	▪ Elicitar ▪ Especificar ▪ Validar ▪ Gestionar	▪ Elicitar ▪ Determinar la solución ▪ Especificar ▪ Mantener	▪ Elicitar ▪ Analizar y Negociar ▪ Documentar ▪ Validar ▪ Gestionar	▪ Elicitar ▪ Analizar ▪ Especificar ▪ Verificar ▪ Gestionar	▪ Identificar ▪ Construir ▪ Organizar ▪ Presentar

Tabla 2-3. Comparativo de Actividades de la IR

En todas las propuestas se comienza elicitando requisitos, que en el caso del IEEE Std P1233 se denomina Identificar Requisitos. Para la actividad de analizar las necesidades de los clientes y usuarios y plasmarlas en modelos o documentos de requisitos, se utilizan en la literatura diversos nombres desde Analizar, Modelar, Especificar, Determinar la solución, Construir requisitos, entre otros. Algunos autores separan la actividad construir modelos de generar el documento de definición de requisitos, tal es el caso de Davis quien las denomina respectivamente: Determinar la Solución - Especificar; el caso de

[19] Se debe hacer notar que IEEE Std 1233/D3, menciona 4 sub-procesos para desarrollar un documento de especificación de requisitos, como producto final y formal del proceso de la IR.

Kotonya & Sommerville: Analizar y Negociar – Documentar; y el caso Thayer & Dorfman: Analizar – Especificar.

Como se puede observar en la Tabla 2-3, el término Analizar tiene diferentes acepciones. En el caso de Leite, denomina Analizar a la actividad de determinar la calidad de los requisitos mediante la verificación y la validación de los modelos y documentos generados. Para Kotonya & Sommerville, Analizar involucra construir modelos y verificarlos. Para Thayer & Dorfman, Analizar es la actividad de estudiar las necesidades de usuarios y clientes para definir los requisitos del software, es decir modelar los requisitos.

Con respecto al aseguramiento de la calidad de los requisitos, las actividades de verificación y validación están en algunos autores claramente identificadas, ambas o una de ellas, y en otros forman parte de la actividad de modelar / especificar. Como se mencionó en el párrafo anterior, para Leite Analizar es Verificar y Validar, distinguiéndolas como dos sub-actividades separadas. En el caso de Loucopoulos & Karakostas, y de Kotonya & Sommerville, separan la actividad de Validar, mientras que la verificación es parte del modelado / especificación. En el caso de Davis, el Mantenimiento incluye verificación y validación. En el caso de Thayer & Dorfman, al mencionar Verificar se están refiriendo en realidad tanto a verificar como validar. En IEEE Std P1233, la verificación es parte de las actividades Construir y Organizar los requisitos mientras que la validación es parte de Presentar los requisitos.

Con respecto a la Negociación, está claramente diferenciada en Kotonya & Sommerville; en el marco de esta tesis se la incluyó dentro de la actividad de Análisis, mientras que para Leite[20] la negociación es una actividad que se realiza constantemente como parte intrínseca de las restantes actividades de Elicitar, Modelar y Analizar. Tampoco es mencionada como una actividad separada por otros autores. Con respecto a la negociación mencionada por Kotonya & Sommerville, la misma difiere respecto a esta presentación (ver sub-sección 2.2.6) pues para ellos la negociación consiste básicamente en la resolución de conflictos entre clientes y usuarios, para esta tesis incluye además la gestión de las propuestas de requisitos que elaboran los ingenieros.

La gestión de requisitos es considerada hoy en día una actividad crítica en el proceso de la IR, y así lo consideran tanto Thayer & Dorfman como Kotonya & Sommerville. En el caso de Leite no es mencionada directamente en [Leite 94] pero sí posteriormente [Leite 01b] [Leite 04]. En el caso de Davis, considera a la gestión como parte de las 4 actividades de Elicitar, Determinar la Solución, Especificar y Mantener. En el caso de Loucopoulos & Karakostas, hacen mención a las circunstancias cambiantes y a la necesidad de hacer frente a ellas capturando y registrando en todo momento la justificación de las especificaciones en los modelos, para asegurar un adecuado mantenimiento de los modelos y productos de software.

[20] Julio Leite en comunicación privada a Jorge Doorn, Gladys Kaplan y Graciela Hadad en Buenos Aires, Diciembre 2004.

2.3. La Ingeniería de Requisitos en los Modelos de Proceso de Software

2.3.1. Modelos de Proceso de Software

No existe un proceso de desarrollo de software ideal, sino que a lo largo del tiempo diferentes organizaciones han desarrollado enfoques completamente diferentes centrados en diversos aspectos de la producción del software, ya sea: las actividades del proceso, los productos de cada actividad, el tipo de artefacto de software a construir, estándares y políticas de la organización.

Se analizan a continuación los llamados paradigmas de proceso de software, enfocando el presente estudio en el tratamiento que dan a los requisitos.

En el **Modelo de Cascada** (también denominado *Modelo Convencional*), desarrollado por [Royce 70], donde cada fase autónoma estrictamente secuencial produce una salida o documento aprobado, comienza por la fase de Análisis y Definición de Requisitos, obteniendo como salida el documento de Especificación de Requisitos de Software. Este hito en el desarrollo provoca el "congelamiento" prematuro de los requisitos, es decir este modelo provee una visión estática de los requisitos del software. Esto impide una adecuada gestión en los cambios de los requisitos: cuando éstos ocurren durante cualquier etapa posterior del desarrollo, el modelo obliga a que recién al finalizar el proyecto estos cambios puedan ser tenidos en cuenta. Por otro lado, la participación de los clientes y usuarios ocurre exclusivamente en la primera fase de definición de requisitos y en la última etapa de prueba de aceptación del sistema, en oposición a uno de los principios de la IR que es el esfuerzo cooperativo de todos los involucrados durante todo el proceso de desarrollo del software.

El proceso de desarrollo de software es por un lado un proceso de resolución de problemas y por otro un proceso ingenieril (ver sub-sección 1.1.3), pero este modelo presenta una visión estricta de manufactura de un producto.

En 1992, el Ministerio de Defensa alemán produjo una variante del modelo de cascada denominado **Modelo V** [GMoD 92], que se centra en la actividad[21] y en el aseguramiento de calidad, haciendo explícitas la iteración y el rehacer de tareas después de cada prueba específica, pero no resuelve los problemas de evolución de los requisitos ni de colaboración constante de los clientes y usuarios.

En el **Modelo de Prototipos** [Floyd 84] se construye y experimenta con una versión mock-up ("imitación") del sistema de software para comprender la

[21] El Modelo de Cascada se centra en el producto resultante de cada fase del desarrollo.

funcionalidad requerida. El prototipado[22] es la actividad central del desarrollo del software, siendo su objetivo reducir los riesgos en el desarrollo. La elicitación, análisis y especificación de requisitos están afectadas por el planeamiento, desarrollo y prueba del prototipo. Es decir, a partir de una comprensión inicial de las necesidades del usuario se determinan los objetivos y alcances del prototipo. El prototipo se construye sobre la base de requisitos informales e incompletos, que se van ampliando y mejorando a través de la ejecución del prototipo con los usuarios. Cuando el prototipo es refinado hasta lograr la versión final del software, entonces el prototipo, denominado prototipo evolutivo, es en sí mismo la especificación de requisitos. En cambio, cuando a partir del prototipo, se continúa con un desarrollo convencional, se debe generar un documento de especificación de requisitos basado en este prototipo desechable [Gomaa 81].

Este modelo logra una alta participación de los usuarios pero a menudo suele obtenerse un sistema de software con una estructura deficiente, debido a los continuos cambios realizados sobre el prototipo. Esto dificulta la detección de inconsistencias en el diseño y de errores en el código. Pero el principal problema surge cuando los usuarios no comprenden el rol del prototipo y asumen que éste ya es la solución al problema. Entonces los usuarios adquieren la sensación de que el sistema de software está completado pero se ven frustrados por el retraso inexplicable en la puesta en operación del sistema.

En el **Modelo de Especificación Operacional**, desarrollado por [Zave 84], se genera en la etapa inicial del proyecto un modelo ejecutable del sistema, denominado "especificación operacional", el cual permite evaluar el comportamiento del sistema y la estructura del software. Es decir, parte del diseño se realiza prematuramente. La elicitación de requisitos es el paso previo a la construcción de la especificación operacional, y ésta última corresponde a la definición de requisitos pero incluyendo algunas tareas de diseño. La validación de requisitos corresponde al uso / prueba / ejecución del modelo de especificación operacional. El principal inconveniente con este modelo radica en que la especificación operacional no es independiente de la implementación.

En el **Modelo de Transformación Formal**, propuesto por [Balzer 83], hay una elicitación y definición informal e incompleta de requisitos, a partir de la cual se construye una especificación formal de requisitos expresada en notación matemática. Este modelo utiliza una serie de transformaciones para convertir la especificación en un sistema de software concreto. Dado que la especificación formal es difícil de comprender por el usuario, se realizan validaciones de requisitos mediante simulaciones. La construcción de la especificación formal en sí misma y su posterior validación son procesos altamente costosos, que se justifican sólo en sistemas con requisitos severos de protección, fiabilidad o seguridad, y además con requisitos estables [Sommerville 02].

[22] El prototipado es además una técnica que se utiliza en otros modelos de proceso de software, como en el modelo espiral para evaluar riesgos de desarrollo, o para mejorar los problemas del modelo de cascada. También se utiliza como técnica para elicitación de requisitos y como técnica de validación de los mismos.

En el **Modelo Incremental**, propuesto por [Mills 80], inicialmente se define un bosquejo de los requisitos, los usuarios los priorizan y luego, se planifican los sucesivos incrementos, donde para cada uno se indica el subconjunto de funcionalidades del sistema que se entregará. La asignación de funcionalidades a cada incremento depende de la prioridad dada a los requisitos. Cada incremento consta de las siguientes actividades: desarrollar el incremento, validar el incremento, integrarlo y validar el sistema. En la concepción original de este modelo, los requisitos se consideraban estables, pero en la práctica, debido a las entregas parciales, los usuarios experimentan con partes del sistema de software lo que los ayuda a clarificar sus necesidades para siguientes entregas.

En el **Modelo Evolutivo** (también denominado *Modelo Iterativo*[23]), [McCracken 82] [Brooks 87b], donde se entrega un sistema completo desde el principio pero en sucesivas iteraciones se va refinando el sistema hasta lograr la versión final del sistema, ocurre que en cada iteración se realiza la especificación de requisitos, el diseño, la codificación y la validación del sistema. Luego, la especificación de requisitos se realiza en forma creciente: a medida que los usuarios logran un mejor entendimiento del problema, éste es reflejado en el sistema de software. Es decir, cada producto en una etapa de especificación de requisitos es un agregado o mejora al producto de la etapa de especificación anterior. Como desventaja, se presenta que, si los lapsos entre entregas son cortos, puede ser muy costoso producir documentos que reflejen cada versión del sistema.

Este modelo se basa en dos premisas: i) los usuarios a menudo no saben bien lo que quieren o necesitan (ver sub-sección 1.1.2), y ii) casi siempre los requisitos en algún momento van a cambiar (ver sub-sección 1.1.4). Para solucionar el primer punto, los requisitos se determinan en base a alguna forma operacional del sistema (por ejemplo, un prototipo) para ser revisado por los usuarios. Para atender el segundo punto, se realizan entregas parciales del sistema que permiten incorporar nuevos requisitos o cambios en requisitos existentes en la siguiente entrega.

En el **Modelo Espiral**, propuesto por [Boehm 88], que reconoce la naturaleza iterativa del desarrollo, y combina actividades de desarrollo con gestión de riesgo, se introducen dos sub-procesos de la IR que son el análisis de riesgo de los requisitos, mediante técnicas de simulación o prototipado, y la planificación para el diseño. Esto reduce el riesgo de cambio en subsiguientes etapas porque ya se ha evaluado la factibilidad de los requisitos, pero no puede cubrir cambios no previstos ni cuál será su impacto en las siguientes fases, con lo cual se deberán repetir etapas del desarrollo.

En el **Modelo basado en Reutilización** [Jones 84] [Arango 88], que se centra en integrar un gran número de componentes de software existentes, el proceso de IR involucra una primera fase de definición de requisitos, seguida de un análisis de componentes existentes que cubran los requisitos

[23] Algunos desarrolladores consideran al desarrollo iterativo como meramente iterar para realizar correcciones, pero otros en cambio consideran la iteración no sólo como rehacer trabajo sino avanzar en la evolución del desarrollo [Larman 03].

especificados y posteriormente una fase de modificación de requisitos en función de los componentes disponibles; si la modificación no es aceptable entonces se retorna a la fase de análisis de componentes. Debido al reuso de componentes es posible que no se cumplan con las necesidades reales de los usuarios. Por otro lado, la evolución de los requisitos puede requerir cambios en los componentes, que muchas veces no es una opción posible.

2.3.2. Observaciones sobre los Modelos de Proceso de Software

La Figura 2-7 extraída de [Davis 88] compara la satisfacción de las necesidades de los usuarios según los distintos modelos de ciclo de vida. Esta comparación parte de la premisa que las necesidades de los usuarios evolucionan con el tiempo. En un extremo se muestra la función que representa las necesidades de usuarios[24] y en el otro extremo, cómo el modelo de cascada las cubre en cantidad de funcionalidad incluida y en tiempo. Como se observa en dicha Figura, el resto de los modelos están representados por funciones que se acercan más a la función de necesidades que el enfoque convencional; es decir, estos modelos disminuyen:

i) el tiempo para satisfacer una necesidad desde su ocurrencia, y
ii) la distancia entre las funcionalidades incluidas en el sistema en un momento dado y el total de necesidades requeridas hasta ese momento.

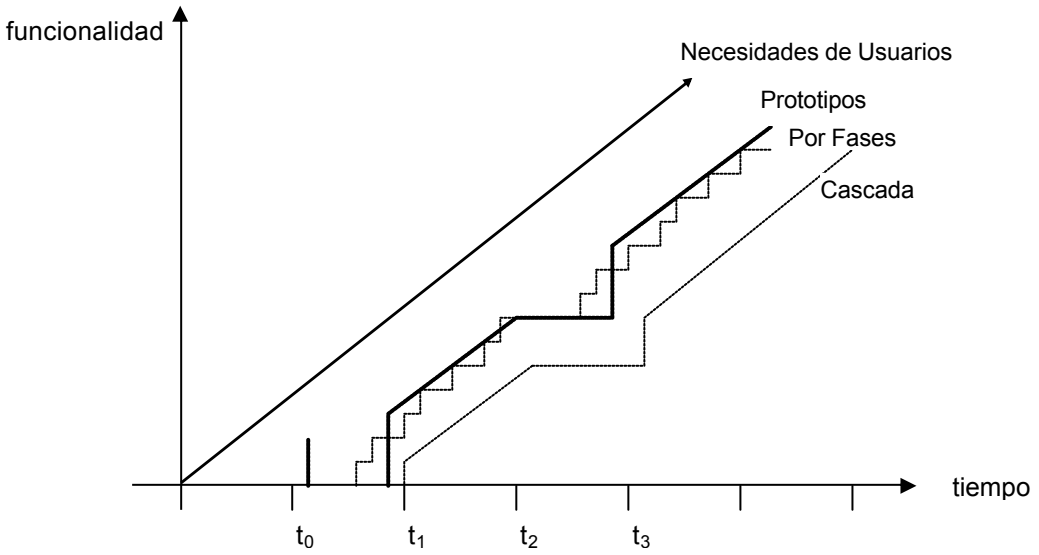


Figura 2-7. Modelos de Proceso de Software versus Necesidades de Usuarios

Los modelos desarrollados con posterioridad al modelo convencional han mejorado de distintas formas los inconvenientes presentados por el modelo de cascada, cuyas consecuencias se han manifestado en los fracasos paradigmáticos mencionados en el capítulo 1. Pero además, de la Figura 2-7

[24] Davis aclara en [Davis 88] que la función de necesidades de usuarios no es realmente lineal ni continua.

se desprende que, aunque algunos modelos de proceso consideran la evolución de los requisitos, ellos no alcanzan a atender en tiempo y forma dicha evolución. Es por ello que nuevos enfoques en el desarrollo de software deben apuntar a obtener resultados parciales del sistema de software que sean más adaptables a la continua evolución de los requisitos. Es decir, deben producir versiones de sistemas de software que atiendan la totalidad de las necesidades a un momento dado y que sean altamente flexibles para satisfacer rápidamente los próximos nuevos requisitos.

Cabe destacar por otro lado que los modelos que más soportan la evolución de los requisitos son aquellos que presentan procesos iterativos, como el modelo evolutivo, el incremental y el espiral, donde la especificación de los requisitos acompaña la implementación del software. Debido a lo cual, no se cuenta con un documento SRS completo final que sirva como contrato para el desarrollo del software. Lamentablemente, esto es una condición mandatoria en muchas organizaciones, principalmente gubernamentales, cuando el proveedor de software pertenece a una organización diferente de la del cliente y existe una relación contractual, formal o informal entre ellas.

Comparativamente, los modelos de prototipado, operacional y espiral, apuntan más a la validación de los requisitos, mediante la producción de un artefacto que les permite a los usuarios experimentar tempranamente en el ciclo de vida.

En resumen, no existe un modelo de proceso ideal de desarrollo de software aplicable a cualquier tipo de sistema, en cualquier tipo de organización, y que garantice el mejor producto a un costo acorde. De ahí que surja una diversidad de métodos, que se basan en algunos de estos modelos o combinaciones de ellos, y que atienden algunos de los subprocesos involucrados en el desarrollo de software. Por ejemplo, el método de “Cuarto Limpio” [Mills 87] integra el modelo de transformación formal con el desarrollo incremental: en cada incremento se desarrolla y valida una especificación formal.

Es también el caso de muchas organizaciones que usan una combinación del modelo incremental con el modelo iterativo (denominado prácticas IID), donde en cada versión se agregan funcionalidades y se mejoran funcionalidades existentes en la versión actual.

Un ejemplo de método basado en desarrollo iterativo e incremental ampliamente difundido es el Rational Unified Process [RUP 98] [Jacobson 99]. Cabe notar que hubo variados antecedentes a él, como por ejemplo: el método EVO propuesto por Gilb en 1988 [Gilb 89], el método Scrum de Sutherland & Schwaber de comienzos de los 90 [Beedle 99], el método DSDM (Dynamic Systems Development Method) desarrollado en 1994 [Stapleton 97], basado en RAD y las prácticas IID, entre otros. Posterior a estos métodos, y en paralelo con RUP, surgieron otros métodos IID, tales como: FDD (Feature-Driven Development) en 1999 creado por DeLuca & Coad [Coad 99] y XP (eXtreme Programming) desarrollado por Beck [Beck 99].

Recientemente se establecieron los "métodos ágiles" cuando en febrero del 2001 se reunieron expertos en DSDM, XP, Scrum, FDD y otros, y formaron la "Agile Alliance" [AgileAlliance 01] [AgileManifesto 01] [Cockburn 02]. Estos métodos ágiles sí tienen en cuenta la evolución de los requisitos, ya que en ellos todo evoluciona, pero no presentan un proceso dedicado a la definición de requisitos sino que éste es más bien informal (ver sección 2.5).

Se debe tener presente que los desarrollos iterativos e incrementales, a pesar de su popularidad actual como piedra basal de los métodos ágiles, y de su puesta en conocimiento a fines de los 80 a través de reportes de resultados de proyectos y artículos científicos, realmente ya habían sido puestos en práctica y con éxito desde la década del 70 en pleno auge del desarrollo en cascada e inclusive se remontan a proyectos aislados desde fines de los 50 [Larman 03]. A pesar de esta popularidad, en un estudio realizado por Neill & Laplante [Neill 03] sobre 194 encuestados con diversas posiciones y experiencias en variadas organizaciones, comprobó que el 35% seguía utilizando el modelo de cascada, seguido por un poco más del 25% que utilizaba el modelo incremental.

En organizaciones con una cultura muy arraigada (generalmente pertenecientes a sociedades con gran acerbo cultural), la gente es una fuente de información muy valiosa por su acceso inmediato y su alta permanencia en la organización, facilitando el reuso de información, mientras que en organizaciones donde no hay una cultura afianzada basada en la gente, el reuso de información tiene un sesgo mucho más tecnológico, debiéndose recurrir a instrumentos portadores de información, como documentación y modelos. Se puede decir entonces que la cultura de una organización tiene un efecto de alta influencia en el proceso de desarrollo de software, y muy en particular en el proceso de definición de requisitos [Kotonya 98, pág.30].

Otro factor relevante en el proceso de desarrollo y mantenimiento del software es la cultura del equipo desarrollador, principalmente cuando se trata de un grupo externo a la organización. Esto es debido a que en grupos de alta rotación de sus integrantes entre distintos proyectos, es indispensable la documentación como medio de transferencia del conocimiento adquirido sobre el UdeD bajo estudio [Ravid 00]. Casos como éstos requieren la producción y mantenimiento de modelos y documentos que representen sin ambigüedad y correctamente los requisitos del software, su diseño y su implementación, influenciando este factor la selección de un proceso de desarrollo basado en modelos, en oposición a aquellos basados en la gente [Highsmith 02b]. Como señalan Kohler & Paech [Kohler 02], a veces, lo difícil es balancear la producción de documentación frente a la calidad de la misma y a su tiempo de gestación.

La Gestión del Conocimiento es una área que podría dar solución a la pérdida de conocimiento en una organización [Rus 02], pérdida debida en gran parte al tipo de cultura de la misma. Frecuentemente, en culturas basadas en el conocimiento tácito o informal, esta pérdida se origina, por ejemplo, en la reducción del personal, el incremento de movilidad del personal, la reducción de tiempos de entrenamiento, entre otras causas. Aunque en culturas basadas

en el conocimiento explícito, la pérdida se debe a que la adquisición de conocimiento a veces no es registrada oportunamente o completamente, ya sea, por falta de tiempo o escasez de recursos humanos o tecnológicos.

2.4. Métodos de la Ingeniería de Requisitos

2.4.1. Introducción

En las sub-secciones siguientes se describen brevemente[25] algunos métodos utilizados para la definición de requisitos. Algunos de ellos son métodos que involucran todo el proceso de desarrollo de software, y no exclusivamente a la IR, en esos casos se hace hincapié en el tratamiento que brinda el método a los requisitos.

Los métodos se han agrupado basándose en sus respectivos paradigmas y se presentan en forma cronológica. Dada la gran difusión actual del Proceso Unificado, éste se describe en una sub-sección separada.

2.4.2. Técnicas tradicionales de Análisis de Sistemas[26]

El **Análisis Estructurado** [DeMarco 78] [Gane 79] es una técnica centrada en los procesos y basada en modelos, que se utiliza para representar requisitos del negocio para un sistema. Esta técnica introdujo el modelo DFD que ilustra los procesos, sus entradas, sus salidas y sus reservorios. El mayor énfasis de esta técnica se coloca en el sistema y su descomposición en subsistemas y funciones a través de niveles de detalle crecientes. Esta técnica comienza construyendo el modelo físico del sistema existente, a partir de ahí deriva el modelo lógico actual, para luego construir el modelo lógico del sistema a desarrollar, y su posterior modelo físico. Esta técnica no involucra a los clientes y usuarios durante el proceso, sólo en la etapa de elicitación de los requisitos, generando un conjunto importante de modelos de requisitos: DFD, diccionario de datos y especificaciones de los procesos primitivos ("mini-specs"). Gane & Sarson [Gane 79] introdujeron una notación diferente a la de Tom DeMarco e incluyeron un diagrama de acceso de datos donde se describe la estructura y contenido de los datos.

El **Análisis Esencial** [McMenamin 84] es una técnica dirigida por eventos, que surge como una mejora al análisis estructurado mediante la generación de una lista de eventos externos al sistema frente al cual el sistema

[25] Por razones de espacio, no se pretende ser exhaustivo sino mostrar métodos relevantes en el área de la IR que permitan encuadrar el tema de la presente tesis.

[26] La mención de Análisis de Sistemas en este texto refiere, en el marco del modelo tradicional de cascada, a la etapa inicial previa al diseño del software, cuya definición en [Senn 89] dice: "es el proceso de clasificación e interpretación de hechos, diagnóstico de problemas y empleo de la información para recomendar mejoras al sistema". Una versión más "aggiornada" de esta definición es la dada en [Whitten 03]: "es el estudio del dominio del problema para recomendar mejoras y especificar los requisitos de una solución". Como señala Alejandro Oliveros en [Oliveros 04], el Análisis de Sistemas tenía un alcance más acotado respecto a la IR, pues adolecía de ciertos aspectos como el tratamiento de RNF y la aplicación sistemática de las actividades de elicitación y validación de requisitos.

debe responder. Esta visión tiende a enfatizar una concepción del proceso del negocio como un sistema reactivo que no siempre es consistente con la realidad organizacional. Esta técnica refuerza la distinción entre los modelos de procesos lógicos y físicos, dando mayor importancia a los modelos lógicos, que denomina modelos esenciales. Basado en esta técnica, surge el Análisis Estructurado Moderno de [Yourdon 89] que abandona el análisis estructurado clásico que comenzaba construyendo modelos físicos del sistema actual y formaliza el enfoque dirigido por eventos. En el análisis estructurado moderno se introdujo también el modelado de datos.

La **Ingeniería de Información** [Martin 90] es una técnica basada en modelos pero centrada en los datos aunque sensible a los procesos, que se utiliza para modelar requisitos y diseñar sistemas. Utiliza como modelo principal el DER. Esta técnica primero se enfoca en modelar los datos a través de los DER, y luego utiliza DFD y diagramas de estructura para modelar los procesos del sistema que usan esos datos. La Ingeniería de Información modela los requisitos de datos de la organización como un todo, en oposición a modelar un solo sistema de información de la organización. Esta técnica cubre casi todo el ciclo de vida, excepto el soporte de sistemas. Las fases que cubre son: i) la planificación estratégica de sistemas para la organización, ii) el análisis de áreas de empresa (subsistemas), iii) el análisis y diseño de aplicaciones para una área de empresa (aplicando técnicas estructuradas), y iv) la implantación de la aplicación diseñada.

Tanto el análisis estructurado como la ingeniería de información tratan de sincronizar los modelos de proceso y los modelos de datos. La diferencia básica entre ambas técnicas es cuál modelo construyen primero: el análisis estructurado comienza construyendo modelos de proceso mientras que la ingeniería de información primero construye modelos de datos.

El paradigma de estas técnicas estructuradas se basa en estudiar el problema y dividirlo en problemas menores hasta llegar a fragmentos fácilmente entendibles. Este enfoque usualmente denominado top-down para la comprensión de un problema tiene una dificultad mayor: lograr la claridad para dividir apropiadamente el problema que está bajo estudio y que, por lo tanto, se desconoce [Jackson 95]. Adicionalmente, estas técnicas se centran en modelar el problema desde el punto de vista del sistema a construir, sin elicitar previamente información del contexto en el cual dicho sistema se utilizará. Es entonces, que se añade otra dificultad a la división del problema: modelar una solución que puede no ser la adecuada por no comprenderse apropiadamente el UdeD.

Observando estas técnicas desde la perspectiva actual de los nuevos conocimientos en requisitos y de software en general, se puede mencionar que en estas técnicas los requisitos están empotrados en los procesos, no habiendo heurísticas que determinen cómo deben explicitarse.

Como se verá en los capítulos 6 y 7 (tratamiento de escenarios actuales y escenarios futuros), las técnicas estructuradas también diferencian la realidad preexistente de la realidad imaginada con el nuevo software en funcionamiento,

pero no dan guías para explicitar los requisitos, los que permanecen empotrados en los modelos o "mágicamente" aparecen en los documentos de especificación de requisitos.

Por otro lado, estas técnicas se centran básicamente en el modelado, haciendo uso de técnicas primitivas para la recolección de datos, sin profundizar en su aplicación, y con escasas propuestas para la V&V de los modelos construidos.

2.4.3. Métodos Orientadas a Objetos

Las técnicas de análisis y diseño orientadas a objetos también están basadas en modelos pero a diferencia de las técnicas estructuradas y de ingeniería de información que tratan por separado los procesos y los datos, integran estos dos bloques bajo el concepto de objetos e introducen los diagramas de clase que ilustran un sistema en término de sus objetos y sus interacciones. Estas técnicas se basan en la idea que los objetos existen en el entorno del sistema.

En el **enfoque de Coad & Jourdon** [Coad 91], el análisis orientado a objetos consiste en 5 capas: i) clase-objeto, ii) estructura, iii) asunto, iv) atributo y v) servicio. El desarrollo de cada capa agrega información a la especificación resultante. La capa clase-objeto consiste en identificar las clases y objetos claves de la aplicación. En la capa estructura, se describen las relaciones estáticas entre objetos: agregación y herencia. La capa asunto consiste en agrupar objetos o clases en áreas mayores (parecido al concepto de subsistemas), esto permite dar una visión de nivel alto del sistema. Las capas siguientes atributo y servicio es donde se describen en detalles los objetos y se identifican otros tipos de relaciones. En la capa de servicio se modela cada objeto como una máquina de estado para identificar servicios, y a partir de ahí se identifican los mensajes entre objetos. Esta visión trata a los objetos como elementos reactivos cambiando de estado frente a eventos, aunque no siempre los elementos del negocio actúan de esa manera. Se generan diagramas de servicio para especificar la lógica procedural de cada servicio.

El **método OMT** [Rumbaugh 91] abarca las siguientes etapas: i) el análisis, ii) el diseño del sistema, iii) el diseño de objetos y iv) la implementación de la solución, usando los mismos conceptos y la misma notación a lo largo del proceso. El análisis produce un modelo de objetos encontrados en el dominio de la aplicación. Durante el diseño del sistema, se define la arquitectura del sistema y éste se organiza en subsistemas. En el diseño de objetos, al modelo de análisis se le agregan detalles de implementación. La implementación consiste en la programación de los objetos, dando guías para realizarla en distintos entornos: con lenguajes orientados a objetos, con lenguajes estructurados y con bases de datos relacionales. El método emplea una notación gráfica propia para expresar los tres tipos de modelos que utiliza en el proceso. El modelo de objetos describe la estructura de los objetos y sus relaciones, y contiene un conjunto de diagramas de objetos. El modelo dinámico describe las interacciones entre

objetos a través de diagramas de estado; y se vale de escenarios y diagramas de secuencia para construir los diagramas de estado. El tercer modelo funcional describe las transformaciones de datos del sistema utilizando DFD. En el análisis se parte de un manifiesto de los clientes, se realizan entrevistas y, una vez adquirido el conocimiento sobre el dominio de la aplicación, se construyen los tres modelos, los cuales van evolucionando a lo largo del proceso de desarrollo, agregándoles detalles en cada etapa. Como indican los autores, la notación utilizada en el modelo de objetos deriva de la utilizada en el DER de [Chen 76], y los diagramas de estado derivan de la notación de [Harel 87]. Además, cabe notar que gran parte de la notación de los modelos de objetos y dinámico se incorporó posteriormente a UML.

El **método dirigido por responsabilidades** propuesto inicialmente por Wirfs-Brock et al. [Wirfs-Brock 90] se abocaba al diseño orientado a objetos, pero luego evolucionó para incluir también el análisis [Wirfs-Brock 95]. El proceso comienza con la fase de descripción del sistema que consiste en comprender a los clientes y usuarios del sistema, identificar los actores (quiénes interactúan con el sistema: personas u otros sistemas), describir escenarios de operación del sistema y construir "conversaciones" entre el sistema y los actores. La fase exploratoria, que es la parte original del método, consiste en identificar clases, responsabilidades y colaboraciones, que se describen en tarjetas. En la fase de refinamiento, se desarrolla la jerarquía de herencia de clases y se detallan las relaciones entre las clases. En este método se especifica en cada clase no sólo los servicios que la clase proveerá sino también los servicios que usará de otras clases (colaboradores) para cumplir con sus responsabilidades. Este método utiliza gráficos de colaboración para representar los contratos entre clases clientes y clases servidores, gráficos de jerarquía para representar clases y subclases, y diagramas de Venn para expresar responsabilidades comunes de clases. Desde el punto de vista de satisfacer los requisitos funcionales (RF) (ver sub-sección 2.6.4) del sistema, este método pone énfasis en el estudio de las interacciones entre clases, mediante las responsabilidades y colaboraciones.

En los métodos orientados a objetos los requisitos se encuentran empotrados en los objetos del software. Estos métodos estudian el UdeD a través de modelos de objetos, por lo tanto, omiten comprender el UdeD con su propia problemática, objetivos, conflictos y expectativas de evolución. Esto no implica necesariamente un empobrecimiento de la comprensión del UdeD, pero tiene un mayor riesgo de que ello ocurra. Estos métodos, en general, enmascaran o subordinan lo que antes se denominaba "análisis"[27] en el diseño, dando mayor predominancia a establecer una solución técnica antes que una solución basada en requisitos. En el caso de Rebeca Wirfs-Brock, se puede decir que conserva esta perspectiva orientada a objetos pero con mayor interés en la comprensión del UdeD.

Al igual que las técnicas estructuradas, estos métodos se centran en el modelado y la notación de sus modelos, dando pocas pautas para la recolección de datos y el análisis de los modelos, y con gran despreocupación

[27] Ver Nota al Pie previa sobre Análisis de Sistemas.

en el pasaje de modelos de análisis a modelos de diseño.

2.4.4. Comparando técnicas estructuradas vs orientadas a objetos desde la IR

Los defensores de desarrollos orientados a objetos alegan que pensar el mundo en términos de los objetos que componen el mundo y sus interacciones facilita el modelado de ese mundo pues es la forma "natural" de pensar acerca del mismo, en oposición a la postura estructurada de pensar en términos de funciones o procesos que ocurren en ese mundo. Sin embargo, los promotores de desarrollos estructurados consideran que su forma de ver el mundo, donde hay un problema complejo que se puede descomponer en problemas menos complejos, también es "natural". Ward & Mellor [Ward 85] sugieren que al modelar un problema en particular se determine la conveniencia de una perspectiva orientada a objetos o estructurada. Tal vez, la mejor solución pase por una tercera alternativa donde la definición de requisitos no sea dependiente de estos dos paradigmas, y que una vez conocido realmente el problema se pueda determinar la mejor estrategia a seguir en las etapas siguientes del desarrollo. En la presente tesis se presenta una tercera alternativa en la IR independiente del problema a estudiar y del método de desarrolla a seguir (ver capítulo 3). En el estudio realizado recientemente sobre la práctica en IR [Neill 03], se observó que las técnicas orientadas a objetos no son tan dominantes como se suponía, y que mientras el 50 % utilizaron escenarios y casos de uso, sólo el 30% utilizó técnicas de análisis orientado a objetos, considerando que estas técnicas se aplican junto con casos de uso (ver sub-sección 2.4.6).

Cabe notar que en ambos paradigmas hay una fuerte interacción entre el modelo de representación preconizado y la técnica de elicitación a utilizar. Mientras las técnicas orientadas a objetos promueven la observación del mundo con sus elementos, las técnicas estructuradas aconsejan la búsqueda de métodos y procedimientos a seguir, plasmados principalmente en la documentación de la organización.

Tanto el análisis estructurado (y sus derivados) como el análisis orientado a objetos enfatizan la generación de diferentes modelos que representen los requisitos desde distintas perspectivas, centrándose en los RF, pero sin considerar apropiadamente todas las actividades que debe abarcar un buen proceso de definición de requisitos. Es acá donde se muestra la brecha entre lo que se conocía como "Análisis de Sistemas" y lo que actualmente se presenta como "Ingeniería de Requisitos".

2.4.5 Otros métodos en la IR

SCR (Software Cost Reduction) [Heninger 78] [Heninger 80] es un método formal para especificar y analizar requisitos de software de sistemas reactivos, que fue posteriormente refinado [van Schouwen 93] [Heitmeyer 95] dándole una semántica formal y herramientas de soporte. Este método se centra en las salidas del sistema, expresa cada salida como una función

matemática del estado e historia del entorno, y especifica el valor de cada salida mediante una notación tabular. La versión más actual del método provee además simulación asistida por herramientas y verificación formal del modelo. Este método se aplica principalmente en el desarrollo de software de tiempo real y de sistemas de seguridad críticos.

En SCR los requisitos del sistema son representados a través de dos modelos: el Modelo de las Cuatro Variables y el Modelo Formal de Requisitos. El primero brinda una descripción abstracta de alto nivel del comportamiento del sistema mediante un conjunto de relaciones matemáticas entre cuatro tipos de variables: monitoreadas (valores del entorno que influyen en el comportamiento del sistema), controladas (valores del entorno que el sistema controla), de entrada (valores resultantes recibidos de dispositivos de entrada según las variables monitoreadas) y de salida (valores entregados a dispositivos de salida para determinar las variables controladas). Este modelo es independiente de implementación, mostrando el comportamiento externo requerido del sistema frente al ambiente. En base a este modelo, se construye el Modelo Formal de Requisitos, que es un modelo de notación tabular que representa al sistema como una máquina de estado, apoyado por un conjunto de definiciones y construcciones [Heitmeyer 96]:

- ⇨ Diccionario de variables (monitoreadas, controladas, de entrada, de salida)
- ⇨ Diccionario de constantes
- ⇨ Diccionario de tipos de datos
- ⇨ Diccionario de clases de modo (una clase de modo es una máquina de estado definida sobre una variable monitoreada)
- ⇨ Diccionario de suposiciones (restricciones acerca del comportamiento del entorno)
- ⇨ Diccionario de afirmaciones (propiedades acerca del comportamiento del sistema)
- ⇨ Tablas que definen la dependencia de variables: tablas de condiciones, tablas de eventos y tablas de transición de estados.

Los pasos a seguir en el método SCR son: i) especificar con precisión el sistema mediante una notación tabular con una semántica formal explícita (se construyen los dos modelos antes mencionados), ii) verificar la especificación usando la técnica de "consistency checking" (se chequean errores de tipo, omisiones, ambigüedades, circularidades, entre otros), iii) simular el comportamiento del sistema (validación), iv) verificar la especificación usando la técnica de "model checking", y v) verificar la especificación usando la técnica "theorem proving". El proceso de especificación (1º paso del método) consiste a su vez en: 1) identificar las salidas del sistema (cantidades controladas), 2) determinar las entradas que el sistema monitorea (cantidades monitoreadas) para producir las salidas, 3) definir variables auxiliares que representan funciones de estado, 4) especificar el comportamiento ideal del sistema (se asume que el sistema puede medir y calcular perfectamente), 5) y especificar el comportamiento real del sistema (se incorporan variables de entrada y de salida). En general, se construye la especificación SCR partiendo de un documento SRS escrito en LN. Este método no incluye la actividad de

elicitación, sino que se basa fuertemente en el modelado, la verificación y la validación, enfatizando la comprensión de la interacción entre el sistema y su entorno, pero sin profundizar en las características intrínsecas del UdeD. Los requisitos son descriptos previamente a la aplicación del método, no presentándose heurísticas que indiquen la actualización / corrección del documento inicial SRS una vez aplicado el método.

El **método del Cuarto Limpio** ("Cleanroom") [Mills 87] [Prowell 99] es un enfoque que integra métodos formales a lo largo del ciclo de vida del software. Las características de este método son: i) Especificación formal mediante un modelo de transición de estados que muestra la respuesta del sistema a estímulos, ii) Desarrollo incremental del software, los incrementos se definen en la etapa inicial del proceso, iii) Programación estructurada: para generar código se aplican transformaciones que preservan la exactitud de la especificación, iv) Verificación estática del software mediante inspecciones rigurosas de código, y v) Pruebas estadísticas del software integrado en cada incremento para certificar su confiabilidad, las pruebas se basan en la especificación formal.

Las etapas del proceso incremental del Cuarto Limpio: i) Establecer los requisitos, generando un documento de definición de requisitos orientado al cliente, ii) Generar la especificación matemática en base al documento anterior, iii) Desarrollar el incremento de software: programar, inspeccionar y probar estadísticamente el código generado, y iv) Entregar el software, con petición de cambios en los requisitos, que retroalimentan el proceso.

Una especificación formal puede reducir los defectos por permitir una representación no ambigua y precisa de los requisitos, y por facilitar la detección de inconsistencias tempranamente, permitiendo generar software con muy pocos defectos, lo que justifica su uso en el desarrollo de sistemas críticos donde es de alta prioridad la seguridad, fiabilidad y protección. Por otro lado, los métodos formales requieren de un equipo de trabajo altamente capacitado y hábil, y del uso de tecnología avanzada, tanto para la generación de la especificación formal como para la ejecución de procesos de V&V. Esto provoca que el costo de desarrollo sea relativamente alto frente a otros enfoques convencionales, pero con bajos costos de mantenimiento correctivo por producir software casi libre de defectos. Desde la perspectiva de los requisitos, los métodos formales pueden probar que un software satisface una especificación formal pero no pueden probar que satisfacen las necesidades y demandas de los clientes y usuarios, con lo cual sigue siendo necesario que el ingeniero de requisitos adquiera una exhaustiva comprensión del UdeD, aún cuando se haya observado en proyectos reales que las especificaciones tienen menos errores [Vienneau 93].

JAD [Wood 89] [Andrews 91] es una técnica orientada especialmente a dar soporte y acelerar la fase de definición de requisitos y la toma de decisiones en el diseño de una solución, y centra su interés en las personas, enfatizando el desarrollo participativo entre clientes, usuarios y desarrolladores. Realiza sesiones intensivas de trabajo estructuradas, formales y focalizadas donde involucra tanto a clientes, usuarios, desarrolladores, la gerencia,

expertos del dominio y expertos en tecnología informática. Esta técnica fue desarrollada originalmente para definir requisitos de alta calidad a través de las sesiones de trabajo, pero dada su creciente popularidad se la comenzó a utilizar en otras fases del desarrollo de software con diversos propósitos, como por ejemplo, especificar conjuntamente opciones técnicas y diseños de interfaces, es por ello que algunos autores definen a JAD como un método de desarrollo de software [Whitten 03]. Hoy en día, esta técnica se la utiliza también en el planeamiento estratégico de la organización, en planes estratégicos de sistemas de información, en la definición de arquitectura de sistemas, en la reingeniería de los procesos del negocio, en el diseño detallado de sistemas, en el modelado de procesos y datos, y en la gestión de proyectos.

Como consecuencia de aplicar esta técnica, se obtienen tres documentos: i) la guía de definición gerencial, que describe el proyecto, los objetivos, alcance y responsabilidades, ii) el documento de trabajo, que describe la información tratada en la sesión, los participantes, suposiciones, requisitos y temas abiertos, y iii) el documento final, que describe las conclusiones finales obtenidas por el grupo, con la definición de los requisitos del sistema. Las cinco fases de JAD son: i) definición del proyecto JAD, ii) investigación sobre los requisitos de usuarios, iii) preparación de la sesión, iv) realización de la sesión, y v) confección y aprobación del documento final que incorpora todas las decisiones tomadas.

Esta técnica ayuda a identificar con mayor rapidez el vocabulario del dominio de la aplicación, los problemas, las necesidades, las prioridades y las soluciones alternativas. JAD reduce el tiempo de definición de requisitos y mejora la calidad de éstos, pues, por un lado, evita las demoras originadas muchas veces por el uso de otras técnicas de elicitación como entrevistas, y, por otro lado, a partir de las sesiones ya se obtiene información consensuada y validada por todos los participantes. Esta técnica se centra en la comprensión rápida e intensiva del software a construir, sin un interés especial en el estudio del UdeD.

RAD [Martin 91] es un método de desarrollo que va desde la elicitación de requisitos hasta la implantación del sistema (no incluye el mantenimiento), y combina técnicas estructuradas, especialmente ingeniería de información, con técnicas de prototipado y la técnica de sesiones JAD para acelerar el desarrollo del software. Su objetivo es lograr un sistema de alta calidad, rápidamente y a bajo costo, siendo la definición de calidad que utiliza: «*satisfacer los verdaderos requisitos del negocio tan efectivamente como sea posible al momento en que el sistema entra en operación*» [Martin 91]. Este método, considerado un ejemplo de modelo espiral [Whitten 03], enfatiza la participación de los usuarios en la construcción rápida y evolutiva de prototipos del sistema, para acelerar el proceso de desarrollo. Las ideas básicas de RAD son: involucrar más activamente a los usuarios en las actividades de análisis, diseño y construcción; organizar el desarrollo en una serie de workshops intensivos (técnica de grupos focalizados) donde participan clientes, usuarios, analistas, desarrolladores y programadores; acelerar las fases de requisitos y diseño mediante un enfoque de construcción iterativo; y reducir el tiempo de entrega de un sistema andando. Requiere el uso de herramientas integradas CASE

para modelar y diseñar el sistema, y para generar código a partir del diseño validado, e intensifica el reuso de componentes de software.

Las etapas del método RAD son: i) planeamiento de requisitos, ii) diseño de usuario, iii) construcción e iv) implementación. La fase de planeamiento de requisitos consiste en un estudio de las áreas inmediatas asociadas al sistema a construir, produciendo una definición de requisitos en términos de la funcionalidad esperada del sistema. Se generan bosquejos de modelos de procesos y de datos, una definición del alcance del sistema y una justificación de costos. Es decir, se construyen unos pocos modelos con menor nivel de detalle respecto a los producidos en enfoques dirigidos por modelos. La fase de diseño de usuario consiste en un estudio detallado de las actividades del negocio relacionadas con el sistema. Se realizan sesiones de grupos focalizados y se crean diagramas de acción que definen las interrelaciones entre procesos y datos. Luego se diseña el sistema en términos de procedimientos y bosquejos de pantallas. Se construyen prototipos y se revisan con los clientes y usuarios. La idea principal es que los usuarios clarifiquen los requisitos a través del uso del prototipo (ver sección 2.3: Modelo de Prototipado). Usa la técnica de "timeboxing" para limitar los ciclos de prototipado y manejar los cambios en los requisitos. En la fase de construcción, un equipo pequeño de desarrolladores trabaja con los usuarios finalizando el diseño y construye el software. Esta etapa itera en los pasos de diseño-codificación interactuando con los usuarios para ajustar los requisitos y probar el software. También se genera documentación para operar el sistema. La fase de implementación consiste en instalar el nuevo sistema, administrar el pasaje del sistema existente al nuevo y realizar la prueba de aceptación de usuarios.

RAD es una técnica apropiada cuando hay incertidumbre en los requisitos o éstos son imprecisos. Se utiliza cuando la eficiencia y la confiabilidad no son factores críticos del sistema y los proyectos tienen un alcance limitado y un objetivo bien definido. Debido al prototipo, se detectan más tempranamente defectos en los requisitos. El cambio es un factor esperado bien manejado por el enfoque iterativo. Pero, por otro lado, como se reduce el análisis del problema, el prototipo puede estar resolviendo el problema equivocado o no estudiando otras soluciones alternativas. Además como RAD enfatiza la rapidez, se puede perder calidad en el producto. Los requisitos están empotrados en el prototipo, con las limitaciones que ello conlleva. El paradigma de este método es concentrar la captura de requisitos en una serie de reuniones de trabajo, estableciendo en ellas casi en simultáneo los requisitos y su diseño técnico.

La **técnica i*** [Yu 95] [Yu 97] se utiliza para modelar los requisitos de la fase temprana[28] de la IR, es decir, es una técnica enfocada al modelado del negocio, mediante la descripción de las razones y motivaciones de los actores involucrados en un ambiente organizacional. Esta técnica permite entonces comprender los objetivos de la organización, modelando las relaciones de los actores, quienes dependen entre ellos para lograr objetivos, realizar tareas y proporcionar recursos. Se basa en la construcción de dos modelos: el Modelo

[28] Yu considera que el proceso de IR consta de dos fases, una fase temprana para establecer los objetivos del sistema y una fase tardía para definir los requisitos del sistema [Yu 97].

de Dependencias Estratégicas y el Modelo de Razones Estratégicos. El primer modelo describe las relaciones entre los actores de la organización, capturando sus intenciones y motivaciones que están detrás de sus actividades. Se definen cuatro tipos de dependencias entre actores: dependencias de objetivos, dependencias de tareas, dependencias de recursos y dependencias de "softgoal"[29]. El Modelo de Razones Estratégicos permite representar con más detalle las razones asociadas con cada actor y sus dependencias. Es decir, brinda una visión interna de los actores, mostrando los elementos intencionales: objetivos, tareas, recursos y "softgoals".

El enfoque i* tiende a considerar el modelado organizacional como una tarea previa para facilitar el modelado de los requisitos del software, permitiendo a su vez el rastreo de éstos a los requisitos del negocio [Loucopoulos 95]. El modelado de los objetivos del negocio aporta varios beneficios: conduce a incorporar requisitos que soporten los objetivos, justifica y explica la presencia de requisitos que de otra manera no sería obvio para los involucrados, puede usarse para asignar responsabilidades a los actores en el sistema, y puede proveer la base para detectar y resolver conflictos que surgen por múltiples puntos de vista entre los clientes y usuarios [Loucopoulos 95]. Los modelos i* son muy ricos pero muy complejos debido a todas sus posibilidades, especialmente el modelo de razones estratégicas, y por lo tanto, difíciles de construir [Leite 04c]. Son modelos muy subjetivos en el momento de construir, se puede tener una idea de cualidad de los modelos pero éstos no tienen capacidad de análisis [Leite 04c]. Se ha combinado el uso de estos modelos como parte de otros métodos, por ejemplo, derivándolos de diagramas de la teoría de actividad [Cruz Neto 04], o aplicándolos intensivamente en el enfoque TROPOS [Castro 02].

Esta técnica es propia de la IR, abarcando una etapa inicial: el modelado del negocio. El paradigma de esta técnica son los objetivos del negocio; los requisitos del negocio están empotrados en ambos modelos, sirviendo para el rastreo hacia atrás desde los requisitos del software a sus orígenes en el negocio y su justificación.

La **técnica de Escenarios** es también propia de la IR y se basa en el paradigma de situaciones. Esta técnica permite describir tanto el UdeD actual como el UdeD imaginado con el nuevo sistema de software. Los escenarios que describen las situaciones del UdeD imaginado capturan los requisitos del software; algunos autores hacen explícitos los requisitos en algún documento mientras que otros los trasladan implícitamente a modelos de objetos, donde en general permanecen empotrados. Se estudia más en detalle esta técnica en la sección 2.8 y, como tema central de esta tesis, en los capítulos 5, 6 y 7.

2.4.6 La IR en el Proceso Unificado

RUP [Jacobson 99] [Kruchten 04], definido por James Rumbaugh, Grady

[29] Softgoal: objetivo cuyo criterio de satisfacción o cumplimiento no está definido con precisión a priori, y está sujeto a un modo de satisfacción establecido por los involucrados [Simon 69].

Booch e Ivar Jacobson (conocidos como "los 3 amigos") junto con otros colaboradores, es un proceso completo de desarrollo de software basado en el modelo iterativo e incremental (ver sección 2.3), dirigido por casos de uso (ver sección 2.8) y centrado en la arquitectura. Por tanto, un proyecto de software que utilice este método se divide en partes más pequeñas o mini proyectos, donde cada mini proyecto es una iteración que resulta en un incremento del producto y cada iteración es controlada reduciendo el riesgo. Los casos de uso [Jacobson 92] se emplean en las cuatro fases del proceso, y la arquitectura del software es considerada parte de las necesidades de los usuarios y, por ello, es capturada en los mismos casos de uso. Es entonces, que tanto la arquitectura como los casos de uso evolucionan en paralelo. Cabe aclarar que realmente RUP [Kruchten 04] es un método más específico que el descripto por "los 3 amigos" en [Jacobson 99][30].

RUP enfatiza el desarrollo y mantenimiento de modelos, usando para ello el lenguaje de modelado UML [Booch 99] [Rumbaugh 05], el cual fue adoptado en 1997 por OMG como un estándar para la notación orientada a objetos.

La gestión de requisitos es una de las seis mejores prácticas que promueve RUP. Ello significa que este proceso describe cómo elicitar, organizar y documentar la funcionalidad requerida y las restricciones [Kruchten 04]. Para ello utiliza casos de uso y escenarios, los cuales capturan los RF y algunos RNF, guían el diseño, la implementación y la prueba del software, y sirven, por ende, para el rastreo a través del desarrollo de software y del sistema mismo.

El proceso transcurre a través de varios ciclos, cada uno concluye con la entrega de un sistema mejorado, incrementado y/o actualizado, y a su vez cada ciclo se descompone en cuatro fases consecutivas:

i) *Fase de inicio*: se desarrolla una visión general de los requisitos y restricciones principales del proyecto, se establece el contexto del negocio y se define el alcance del proyecto, se estiman el presupuesto financiero, el plan del proyecto (con iteraciones, fases y cronograma) y riesgos. Se genera el modelo inicial de casos de uso y un glosario inicial del proyecto (el cual puede estar expresado como un modelo del dominio).

ii) *Fase de elaboración*: se analiza el dominio del problema, se establece la arquitectura del sistema y se desarrolla el plan completo del proyecto. Se completa el modelo de casos de uso y se describen en detalle la mayoría de los casos de uso. Se establecen los RNF. A partir de la comprensión del sistema, su funcionalidad y los RNF, se decide sobre la arquitectura. Se tratan todos los posibles riesgos: en los requisitos, tecnológicos, políticos, etc. Generalmente se construyen prototipos evolutivos.

iii) *Fase de construcción*: se desarrollan los componentes, éstos se

[30] Se utilizará en el resto de la tesis el término RUP aún cuando se estuviese hablando del Proceso Unificado.

integran al software y se realizan las pruebas. Se generan los manuales de usuario y una descripción de la versión actual.

iv) *Fase de transición*: corresponde al traspaso del producto de software a los usuarios. Se hace la prueba beta para validar la nueva versión contra las expectativas de los usuarios. Generalmente se hace operar el nuevo sistema en paralelo con el actual. Se realizan ajustes de eficiencia, se identifican y corrigen algunos defectos, se hacen conversiones de bases de datos actuales, se capacita y da soporte a usuarios y administradores, y se completa la documentación para los usuarios.

Desde un aspecto estático, el proceso describe quién hace (trabajadores), qué cosa (artefactos), cómo (actividades) y cuándo (flujos de trabajo), definiendo las actividades como unidades de trabajo asignadas a un trabajador con un propósito claro, mientras que los flujos de trabajo son secuencias de actividades que producen un resultado de valor esperado [Kruchten 04]. RUP describe seis flujos de trabajo de ingeniería y tres de soporte, los que se ejecutan concurrentemente e interactúan entre sí usando los artefactos de los otros flujos. Se observa en la Figura 2-8 la estructura de RUP con sus fases, iteraciones y flujos de trabajo. Se describen a continuación aquellos flujos más ligados a la IR.

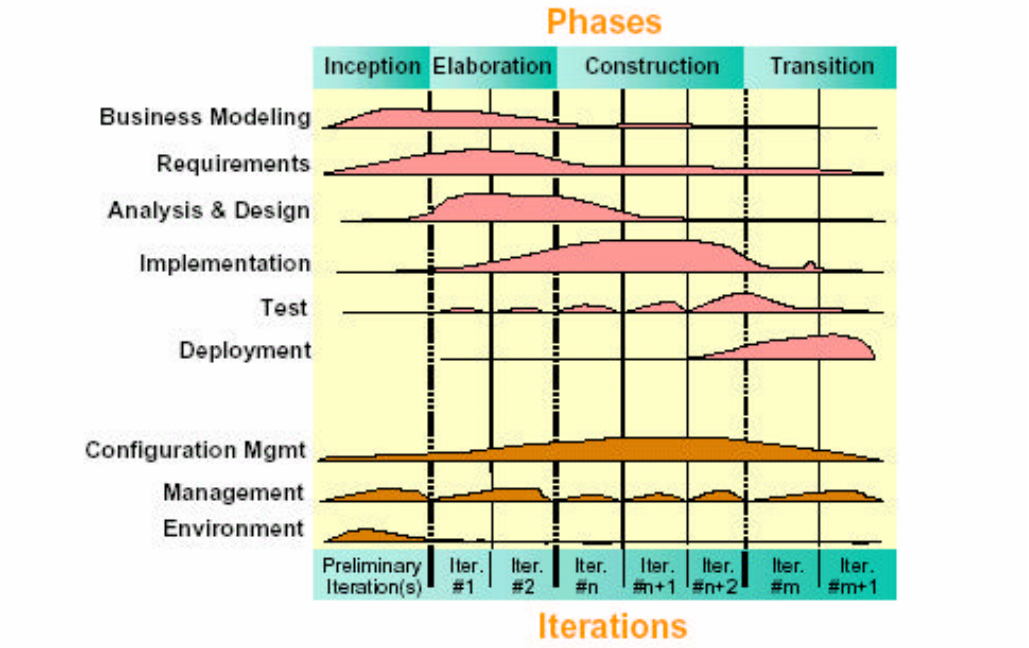


Figura 2-8. Fases y Flujos de Trabajo de RUP

El flujo de Modelado del Negocio se desarrolla principalmente durante las fases de inicio y de elaboración, con escasa actividad durante la construcción y la transición. Mientras que el flujo de Requisitos se desarrolla en forma más pareja a lo largo de las cuatro fases, con mayor carga durante la fase de elaboración.

El flujo de Modelado del Negocio, de ejecución opcional, tiene por

objetivo comprender el contexto del sistema. Maneja dos alternativas de modelado: el modelado del dominio y el modelado del negocio. El primero describe conceptos importantes del contexto como objetos del dominio y sus relaciones, capturados a través de expertos del dominio[31], y representados en un diagrama de clases o un glosario de términos del dominio. El modelado del negocio se utiliza para comprender los procesos del negocio de la organización, manejando dos tipos de representaciones: el modelo de casos de uso del negocio y el modelo de objetos del negocio. Los autores del Proceso Unificado [Jacobson 99, p.118] utilizan los términos "clases del dominio" y "entidades del negocio" en forma indistinta, y aclaran que el modelado del dominio puede considerarse como una variante simplificada del modelado del negocio[32], ya que este último presenta un nivel de detalle mayor, facilitando la traza desde él hacia los casos de uso del sistema.

El flujo de trabajo de Requisitos tiene por objetivo describir qué debe hacer el sistema y permite que los involucrados se pongan de acuerdo sobre dichas descripciones. Se elicitan las necesidades de los clientes y usuarios, y se documentan la funcionalidad requerida y restricciones. Se identifican actores y casos de uso que representan el comportamiento del sistema, creándose el modelo de casos de uso. Se describe en detalle cada caso de uso, cuya descripción muestra cómo el sistema interactúa paso a paso con los actores y qué hace el sistema. Algunos RNF se adosan a los casos de uso y otros se describen en especificaciones suplementarias. El mismo modelo de casos de uso se utiliza tanto durante la captura de requisitos como durante otros dos flujos de trabajo: el de Análisis y Diseño, y el de Prueba.

No es un proceso orientado a objetos sino dirigido por casos de uso, con lo cual tiene una abstracción funcional, la que puede ser considerada una molestia por muchos desarrolladores de hoy en día que emplean técnicas orientadas a objetos y desconfían de una visión funcional.

Este método presenta múltiples ventajas con algunas desventajas, las cuales se señalan en los párrafos subsiguientes.

Dado que se basa en un enfoque iterativo, esto hace que los cambios sean mejor gestionados, los riesgos son considerados tempranamente, tiene un alto nivel de reuso, el grupo de desarrolladores aprende a medida que el proyecto avanza y se evidencia una mejor calidad en general. Como el proceso está centrado en la arquitectura, se gana control, oportunidad de reusar y establece la base para la dirección del proyecto. Al estar dirigido por casos de uso, éstos se expresan desde el punto de vista de los usuarios, son fáciles de comprender al estar escritos en LN, existe un alto grado de rastreabilidad, son la base para la definición de los casos de prueba, y facilitan la planificación de

[31] En la literatura, los modelos generados en el análisis del dominio representan el conocimiento elicitado principalmente de la literatura técnica, implementaciones de sistemas existentes y el consejo de expertos [Arango 89] [Kang 90] [Loucopoulos 95].

[32] La definición de modelado del dominio que utiliza el Proceso Unificado no se corresponde con la expresada por autores que han estudiado el tema de análisis del dominio [Arango 88] [Loucopoulos 95], pues para éstos, un modelo del dominio representa el conjunto de objetos, relaciones y reglas comunes en un dominio y que, por ende, pueden reusarse en diferentes aplicaciones relacionadas a dicho dominio, mientras que un modelo del negocio representa conceptos específicos del dominio de la aplicación.

las iteraciones.

RUP es un proceso completo, bien documentado pero complejo, y esto hace que no sea fácil su aprendizaje y no sea sencillo de aplicar correctamente en una organización, se requiere de una buena capacitación y de gente experimentada que guíe la realización del proyecto.

Es un proceso de desarrollo de software configurable, que puede ser usado tal cual es, tanto en proyectos medianos como grandes, pero es necesario adaptarlo para cada proyecto específico o para la organización. En la mayoría de los casos, se requiere adaptarlo antes de usarlo, tarea nada trivial. RUP describe más de 100 artefactos, lo cual para proyectos pequeños a medianos puede requerir una gestión considerable, es por ello, que este método provee guías para la configuración del proceso (en el flujo de trabajo de Entorno) donde se decide cuáles artefactos se construirán. En el flujo de trabajo de Gestión de Configuración y Cambio, se brindan guías para controlar los ítems de configuración (artefactos) seleccionados.

Dada su complejidad, alrededor de RUP se han definido una serie de procesos simplificados, iterativos e incrementales, tales como el presentado por Fowler en [Fowler 99], GRAPPLE (Guidelines for Rapid Application Engineering) [Schmuller 00], ICONIX [ICONIX 02] [Rosenberg 99] y CUP (ConsultIT Unified Process) [CUP 03].

Pese a lo que señalan sus autores, RUP no atiende los aspectos sociales del desarrollo de software (ver sub-sección 1.1.3) sino que se dedica principalmente a aspectos técnicos de modelado, V&V, arquitectura, diseño e implementación. Es decir, se concentra en describir los artefactos a producir y, cómo y cuándo producirlos, dando pocas pautas para elicitar[33], verificar y validar[34] requisitos.

Se observa que el proceso de desarrollo se centra más en el *diseño de los requisitos para una solución dada* que en estudiar el contexto donde el software operará y en establecer los requisitos del software acorde al negocio e independientes de implementación; pues en los casos de uso que modelan los requisitos ya se incluye la arquitectura del sistema, aún cuando inicialmente se describen casos de uso del negocio cuya utilidad queda diluida en el proceso.

2.5. Métodos Ágiles

2.5.1. Características generales

Los métodos ágiles [Cockburn 02] [Highsmith 02], que son una evolución del modelo iterativo e incremental, están centrados en las personas: le dan un

[33] La descripción de la actividad Encontrar Actores y Casos de Uso en el Flujo de Trabajo de Requisitos en [Jacobson 99] es relativamente vaga sobre cómo detectar fuentes de información, seleccionar técnicas de elicitación y efectuar la recolección misma de información.

[34] La validación de requisitos en el proceso unificado consiste exclusivamente en el prototipado de la interfaz de usuario.

valor preponderante al individuo y al trabajo colaborativo de todos los involucrados, donde los usuarios son parte del equipo de desarrollo. Por otro lado, estos métodos se centran también en el producto de software: su meta es desarrollar un software que esté funcionando en un plazo corto y sin problemas, es decir, se basan en el principio de *"escasa a nula burocracia*" para que no se obstaculice el logro efectivo del software [Highsmith 02b]. Es bajo este foco que se construyen pocos modelos y documentos de soporte al software, y donde se encumbra la cultura de la organización [Highsmith 02b].

Los diferentes métodos ágiles existentes hoy en día no definen un conjunto de pasos estrictos a seguir sino un conjunto de buenas prácticas [Cockburn 02], y se basan en los valores y principios declarados en el Manifiesto Ágil [AgileManifesto 01], siendo el proceso de desarrollo un proceso esencialmente adaptativo, es decir que se lo ajusta a la organización, al tipo de aplicación y a la gente involucrada, y dicho ajuste se realiza durante todo el desarrollo del proyecto [Fowler www]. Es decir, la gran diferencia con los métodos más tradicionales es que estos últimos presentan procesos que prescriben en detalle qué hacer y cómo (denominando a éstos "métodos con procesos predefinidos"), mientras que los métodos ágiles dan recomendaciones y buenas prácticas dependiendo totalmente de la habilidad de los desarrolladores.

Brevemente, las prácticas de los desarrollos ágiles son: iteraciones cortas, pruebas continuas, equipos auto-organizados, colaboración constante y re-planeamiento frecuente basado en la realidad actual.

2.5.2. Relación con los métodos clásicos y la IR

Dadas las características recién mencionadas, estos métodos son eficientes en proyectos con requisitos muy cambiantes, cuando se exige la entrega de un producto rápidamente operativo pero con cierta calidad o cuando están involucradas tecnologías experimentales. La competencia en el ámbito de los negocios exige desarrollos de software que respondan rápidamente a las condiciones cambiantes del mercado. En un contexto de estas características, un método ágil tiene mayores posibilidades de lograr una "solución exitosa" frente a métodos con procesos predefinidos [Highsmith 02b] [Highsmith 01]. Hay que analizar qué se entiende por una "solución exitosa" en los métodos ágiles, si esto significa una solución rápidamente operativa tal que el producto de software se entregue antes que cualquier otro software del competidor del negocio, entonces esto involucra casi inexorablemente una compensación entre agilidad y corrección [Pinheiro 02]. Para Pinheiro, quienes siguen un desarrollo utilizando algún método ágil están entonces considerando al "tiempo de puesta en el mercado del producto" (time-to market) como un requisito mandatorio[35], que puede dejar en un segundo plano algunas otras consideraciones como corrección, propósito, mantenibilidad y escalabilidad entre otras. Además, como se menciona en [Abrahamsson 02, p.99], la evidencia de efectividad y adecuación de los métodos ágiles en distintos

[35] En muchos dominios, time-to-market se ha convertido en una condición primaria para el desarrollo de software, pues éste no puede limitar el desarrollo del negocio [Rockart 96].

ambientes proviene de historias exitosas de profesionales en la práctica, pero existen escasos estudios empíricos que soporten lo dicho.

Por otro lado, para que realmente los clientes y usuarios tengan acceso a un producto en un lapso corto cumpliendo con los objetivos propuestos, se requiere de la participación de uno y, a veces, varios usuarios con dedicación permanente al proyecto. Si esta condición se cumple, la cultura de esta organización es más rápida de obtener y no hay una necesidad imperiosa de generar documentación de soporte al sistema. En un desarrollo de estas características, la fase de mantenimiento del sistema sólo puede sostenerse mediante la cultura, pues existe nula o escasa documentación del sistema. En organizaciones que se basan en la tecnología, generalmente la cultura es una característica de bajo interés, y por lo tanto, recurrir a las personas de la organización no es una opción, es indispensable disponer de documentación del sistema para realizar correcciones, actualizaciones o crecimientos al sistema.

Desde el aspecto social, los métodos ágiles no sólo se basan en la interacción constante con clientes y usuarios sino también en sostener un equipo de desarrolladores estable. Es muy común en ciertas organizaciones la rotación de la gente de sistemas entre proyectos [Ravid 00], pudiendo por ejemplo, el programador no tener acceso durante la implementación al ingeniero de requisitos participante, o aún cuando el desarrollador cumpla ambos roles, puede ocurrir el reemplazo por un nuevo desarrollador sin acceso al anterior miembro. En dichas circunstancias, los métodos ágiles podrían no tener el éxito esperado pues se basan en el conocimiento tácito del equipo de desarrollo [Boehm 02] [Paetsch 03] al minimizar la documentación de los conocimientos adquiridos. Asumen por ello un riesgo que se contrapone con la facilidad de responder rápidamente al cambio.

Los métodos con actividades de IR y diseño logran arquitecturas estables bien concebidas que facilitan el posterior mantenimiento. Los métodos ágiles no ponen énfasis en la arquitectura del software y, por lo tanto, pueden entrar durante la actividad de "refactoring"[36,37] en un permanente estado de mantenimiento, impidiendo el avance en el desarrollo del producto; aún cuando Beck dice en [Beck 00, pág.135] que el mantenimiento es el estado normal de un proyecto XP.

En la práctica, una gran cantidad de proyectos de software cae en el constante mantenimiento sin poder adicionar nuevos componentes al software, donde el presupuesto para desarrollo se destina al mantenimiento debido a los cambios permanentes, no tanto por cambios en el UdeD sino por problemas en la concepción de los requisitos (conflictos, ambigüedades, descubrimiento tardío) o por cambios en los requerimientos de los usuarios al comprender

[36] Se entiende por "refactoring" en los métodos ágiles como la tarea de modificar código en operación de manera que continúe haciendo lo mismo pero de una forma más fácil de entender y mantener [Fowler 99b].

[37] Se utilizará el término en inglés "refactoring" pues su traducción directa "refactorización" tiene un significado totalmente distinto, y de la traducción más cercana "recodificación" no se desprende realmente la actividad involucrada.

mejor ellos mismos el problema visualizado a partir del software entregado. Los métodos ágiles son proclives a caer en este estado continuo de mantenimiento.

Según Dan Berry en [Berry 02], los métodos ágiles hacen que la parte ascendente de la curva de Belady-Lehman (ver sub-sección 1.1.4) sea más empinada y comience antes, debido a prácticas pobres de estos métodos; un ejemplo de éstas es la postergación de "refactoring" por ser una tarea frustrante para los programadores.

Los métodos ágiles realizan entregas rápidas de versiones al no tener que cumplir con ciertas tareas de documentación y de verificación, es decir, mediante la reducción de actividades de IR y de diseño típicas de los métodos con procesos predefinidos.

Según el Manifiesto Ágil [AgileManifesto 01], los métodos ágiles valoran en mayor medida los ítems expresados en la columna ☑☑☑ de la Tabla 2-4, y consideran que los métodos clásicos basados en procesos, por el contrario, promueven los valores de la columna ☒☒☒. Sin embargo, se debe resaltar que la ingeniería de software y, en particular, la IR hacen hincapié en la participación activa de los clientes y usuarios, enfatizando los aspectos sociales que rodean al desarrollo, y la gestión de requisitos en la IR se dedica a responder en forma efectiva a los cambios, ambos valores de la columna ☑☑☑.

Métodos Ágiles

☑☑☑	☒☒☒
▪ Individuos y sus interacciones	▪ Procesos y herramientas
▪ Software en operación	▪ Documentación
▪ Colaboración permanente de los clientes	▪ Negociación de contratos con los clientes
▪ Responder a los cambios	▪ Seguir un plan[38]

Tabla 2-4. Foco de Valores en los Métodos Ágiles [AgileManifesto 01]

En resumen, cabe notar que los métodos ágiles tienen éxito sólo si hay una fuerte participación de los usuarios pues, como se mencionó antes, se basan fuertemente en la cultura de la organización. Adicionalmente, se debe destacar que son métodos que indudablemente tienen en cuenta la evolución de los requisitos, ya que han sido desarrollados especialmente para hacer frente a todo tipo de cambios durante el proyecto, se dice que en ellos "*todo evoluciona*". Aunque manejar los cambios es un aspecto intrínseco de los métodos ágiles, éstos no proveen la trazabilidad de los requisitos desde y hacia sus orígenes en el UdeD y sus implementaciones en el código, debido al escaso soporte que brindan a través de la documentación.

[38] Se debe destacar que algunos métodos ágiles poseen un plan de trabajo, tal es el caso de la Programación Extrema donde una de sus fases es la Planificación (ver sub-sección 2.5.3).

A diferencia de métodos que son adaptables a diversos tamaños de proyectos y de tipos de aplicación, como lo es por ejemplo RUP, los métodos ágiles son fáciles de aprender y de aplicar, pero ofrecen soluciones a medida a proyectos pequeños con equipos de 3 a 20 personas como máximo [Beck 00] y con un entorno de trabajo que facilite la comunicación permanente entre los involucrados.

Un punto de interés común tanto para la IR como para los métodos ágiles es la evolución de los requisitos y su tratamiento. Es en este último tema que ambos divergen, pues los métodos ágiles manejan los cambios directamente adaptando el código mientras que la IR los maneja a través de la trazabilidad de los requisitos, evaluando los cambios mediante el rastreo de los requisitos que cambian, la documentación de soporte a ser actualizada y los componentes de código a modificar. La solución dada por los métodos ágiles es rápida y menos costosa, con poca burocracia pero con poca evaluación, basándose en la actividad de "refactoring" para una mejora posterior del código, pero dificultando el mantenimiento del software cuando está en operación. La solución de la IR probablemente no responda inmediatamente ante la ocurrencia de un cambio en los requisitos, pero sí responde con una evaluación más exhaustiva de los cambios involucrados en todos los productos que acompañan al software, brindando de entrada mejores adaptaciones al código y facilitando el posterior mantenimiento del software en operación.

Los enfoques ágiles asumen que están frente a un UdeD altamente cambiante donde los requisitos son emergentes en vez de definibles tempranamente. Pero estos contextos cambiantes no siempre ocurren (ver sub-sección 1.1.4), muchas veces los requisitos son estables o son críticos; en ambos casos, la aplicación de métodos con procesos predefinidos es más efectiva, pues ellos determinan por anticipado los requisitos, exigiendo además el cumplimiento de sus propiedades inherentes (ver sub-sección 2.6.3).

En resumen, cuando se cuenta con usuarios disponibles a participar en forma continua en un proyecto de software, generalmente en organizaciones basadas en una cultura muy arraigada, es entonces, que los métodos ágiles son extremadamente útiles. Pero, cuando el sistema de software comienza a crecer, debiéndose integrar nuevos módulos constantemente al sistema, requiriendo luego mucha actividad de "refactoring", la eficiencia de los métodos ágiles baja, tornándose más lentos que los métodos tradicionales.

En conclusión, si se puede obtener la participación continua de los usuarios con los programadores y el proyecto no crece en perpetuidad, entonces conviene aplicar los métodos ágiles. Estos indudablemente sirven para entregar un producto cuando hay una urgencia en su uso inmediato, probablemente no obteniendo el mejor producto pero cubriendo la necesidad inminente.

2.5.3. La Programación eXtrema

En la actualidad, uno de los métodos ágiles más populares es la Programación eXtrema (XP) [Beck 00], que se basa en cuatro premisas: i) retro-alimentación continua entre los involucrados, ii) comunicación fluida entre ellos, iii) simplicidad en las soluciones, y iv) coraje para enfrentar los cambios. Este método sostiene un conjunto de 12 prácticas, que ya existían en la ingeniería de software, pero que XP las integra y complementa teniendo en cuenta los aspectos humanos y organizacionales involucrados en el desarrollo del software.

El ciclo de vida de un proyecto usando XP consiste en seis fases [Beck 00]: i) Exploración, ii) Planificación, iii) Iteraciones para la Primera Entrega, iv) Producción, v) Mantenimiento, y vi) Muerte del Proyecto. En la Exploración, los clientes bosquejan aquellas historias de usuario (User-Stories) que son de interés para la primera entrega del software, y los programadores construyen prototipos rápidos explorando las posibilidades sobre la arquitectura del sistema y experimentan con la tecnología que van a usar. En la Planificación, los clientes establecen las prioridades de cada historia de usuario, los programadores estiman el esfuerzo necesario para cada una de ellas y juntos acuerdan un cronograma de la entrega. En la fase de Iteraciones, el cronograma se divide en iteraciones de 1 a 4 semanas cada una. En cada iteración, los clientes producen un juego de casos de prueba funcionales para cada una de las historias programadas para esta iteración, y los programadores trabajando en parejas codifican dichas historias. En la primera iteración se intenta establecer la arquitectura del sistema, mientras que en la última iteración el sistema estará listo para pasar a operación. En la fase de Producción, se realizan pruebas y consecuentes ajustes al rendimiento del sistema, y se decide sobre la inclusión de nuevas características a la versión actual debido a cambios durante la fase. En la fase de Mantenimiento, mientras se realizan nuevas iteraciones, se llevan a cabo tareas de soporte al cliente, se realiza "refactoring", se prueba nueva tecnología para la siguiente versión y se experimenta con nuevas propuestas de arquitectura. La fase de Muerte del Proyecto ocurre cuando el cliente no tiene más historias de usuario para incluir en el sistema. En esta fase, se genera la documentación final del sistema y se examina el sistema para satisfacer necesidades del cliente en cuanto a rendimiento y confiabilidad del software.

Las historias de usuario son la técnica que usa XP para capturar requisitos de software. Estas historias son confeccionadas por los usuarios, donde describen lo que el sistema debe hacer para ellos, o sea, para proveerles "valor de negocio". Luego, las historias se descomponen en "tareas de ingeniería" que se asignan a los programadores para ser implementadas en una iteración. Es decir, casi inmediatamente de las historias escritas por los usuarios se pasa a la codificación, de esta manera se obtiene una solución para el problema concreto sin consideraciones hacia futuras necesidades. Esta "inflexibilidad" en el software se resuelve mediante la actividad de "refactoring" en el momento oportuno, es decir recién cuando se requieren cambios, pero dicha revisión de código y re-diseño se basa indefectiblemente en la suposición de código bien escrito previamente. A partir de las historias los desarrolladores con los usuarios derivan las pruebas funcionales (de aceptación). Estas historias además se utilizan para estimar tiempos de cada iteración.

Existe en la literatura una diversidad de propuestas de templates (fichas) para escribir las historias. Beck [Beck 00] propone como contenido de una historia la siguiente información: fecha, tipo de actividad (alta, corrección o mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con fecha, estado y comentarios. Con las historias de usuarios se presenta la misma duda que se plantea al escribir escenarios y casos de uso cuando no se usan guías o heurísticas: cuál es el nivel de detalle con que se debe describir cada historia. En la literatura actual sobre XP no hay pautas que brinden una respuesta a este tema.

Como se puede observar, desde el punto de vista de la IR, en el método XP los documentos de requisitos corresponden a las historias de usuarios, las cuales son descartadas una vez trasladadas al código, no existe un proceso de definición de las mismas, ni de verificación y/o validación de ellas [Leite 01] [Pinheiro 02]. Es difícil además que los usuarios escriban RNF en dichas historias, sin una capacitación o guía previa al respecto. En la práctica de XP, los RNF son considerados recién al implementar las tareas [Eberlein 02], esto puede ocasionar que realmente la solución adoptada no sea la más adecuada. Por otro lado, el rastreo de los requisitos no es un tema tratado explícitamente en XP debido a la escasa documentación producida [Eberlein 02] [Leite 01], siendo además responsabilidad de los desarrolladores (y no parte del proceso) generar la documentación necesaria para el futuro mantenimiento [Paetsch 03].

La actividad de elicitación en XP se basa en la redacción de historias por parte de los usuarios, y en reuniones entre dichos usuarios y los programadores para recabar detalles sobre dichas historias. Se está asumiendo que aquellos pocos usuarios seleccionados conocen "*todo*" sobre el UdeD y lo saben expresar con claridad y precisión, desestimando otras posibles fuentes de información [Pinheiro 02]; esto es un claro ejemplo de un método basado en la cultura de la organización. Con motivo de la escasez en la variedad y cantidad de las fuentes de información utilizadas, es alta la probabilidad de no ocurrencia de conflictos en los requisitos, no porque éstos no estén presentes en el UdeD sino porque no son detectados al tener una visión parcializada de dicho UdeD. Como se puede notar, XP es altamente dependiente de las habilidades sociales [Kovitz 02] tanto de los programadores para comunicarse con los usuarios como de éstos para transmitir sus conocimientos.

Tanto la IR como XP enfatizan la necesidad de priorizar los requisitos, en XP los propios usuarios dan prioridades a sus historias y también lo hacen los desarrolladores considerando aspectos de implementación [Paetsch 03].

2.6. Los Requisitos

2.6.1. Requerimientos, Requisitos y Especificaciones

Distintos autores en la literatura han contribuido con variadas

definiciones de "Requisito de Software", desde relativamente simples a muy elaboradas. Se presentan a continuación algunas definiciones, comenzando por la que provee la IEEE en el glosario de Ingeniería de Software [IEEE Std 610.12-1990].

> ***Requisito***:
> 1. Una condición o capacidad que necesita un usuario para resolver un problema o alcanzar un objetivo.
> 2. Una condición o capacidad que debe cumplir o poseer un sistema o componente de sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto.
> 3. Una representación documentada de una condición o capacidad como en 1 o 2.
>
> [IEEE Std 610.12-1990]

> ***Requisitos***:
> Descripciones del dominio de la aplicación y de los problemas a ser resueltos en él.
>
> Michael Jackson [Jackson 95, pág.3]

> ***Requisitos***:
> Descripciones de cómo el sistema debería comportarse, de información acerca del dominio de la aplicación, de restricciones en la operación del sistema, o de especificaciones de una propiedad o atributo del sistema.
>
> Kotonya & Sommerville [Kotonya 98, pág.6]

> ***Requisitos***:
> Efectos que la computadora ejerce sobre el dominio del problema, en virtud de la programación de la computadora.
>
> Benjamin Kovitz [Kovitz 98]

> ***Requisito***:
> Todo aquello que debe ser satisfecho, transformado, producido o provisto.
>
> Definición informal de Richard Harwell [Harwell 93]

> ***Requisitos***:
> Determinan lo que hará el sistema y definen las restricciones de su operación e implementación.
>
> Ian Sommerville [Sommerville 02]

> ***Requisito***:
> Una propiedad que debe ser exhibida por un sistema desarrollado o adaptado para resolver un problema particular.
>
> Sawyer & Kotonya [Sawyer 01]

Las definiciones dadas por Jackson [Jackson 95] y por Kotonya & Sommerville [Kotonya 98] llevan a confusión al decir que los requisitos son "descripciones", pues pareciera entonces que están hablando de cómo se

expresa el requisito sin dar realmente una definición del mismo. Ignorando este aspecto, se analiza a continuación las definiciones presentadas y su distinción respecto a la palabra “especificación”.

Tanto la definición de Jackson [Jackson 95] como la de Kovitz [Kovitz 98] se centran en el UdeD, hacen hincapié no en el comportamiento del software sino en los efectos de dicho comportamiento sobre el UdeD. Para ambos, los requisitos se materializan a partir de los fenómenos en el dominio del problema y no del software propiamente dicho. La definición dada por Kotonya & Sommerville [Kotonya 98] es más abarcadora, incluye además al sistema de software. Mientras que tanto la definición informal de Harwell [Harwell 93] como las definiciones de Sawyer & Kotonya [Sawyer 01] y de Sommerville [Sommerville 02] son la visión estricta del sistema. Analizando las tres definiciones de [IEEE Std 610.12-1990], se observa que la primera es la visión de los clientes y usuarios, la segunda definición es la visión desde el sistema de software y la tercera corresponde a la forma en que se expresan dichos requisitos.

Por otro lado, tanto el estándar internacional [ESA PSS-05-0] como Sommerville [Sommerville 02] y Leffingwell & Widrig [Leffingwell 00] hacen una distinción entre requisitos de usuarios y requisitos del sistema. Se debe aclarar que esta distinción realmente no es respecto a los requisitos en sí mismos, sino a la forma de documentar o escribir los mismos. Los requisitos de usuarios son “descripciones” abstractas de alto nivel del comportamiento deseado y útil del sistema, que sirven para comunicarse con los clientes y usuarios, utilizando el lenguaje del usuario. Los requisitos del sistema son “descripciones” detalladas de lo que el sistema va a hacer (sin decir cómo lo va a hacer, ver sub-sección 2.6.2) y sus restricciones expresados en un lenguaje más técnico, que permiten a los desarrolladores escribir código a partir de ellos. Leffingwell & Widrig [Leffingwell 00] denominan a estas dos formas de expresar los requisitos: “Aspectos del sistema” y “Requisitos del Software” respectivamente.

Ian Alexander [Alexander 01] discrepa en cuanto a la utilidad de escribir los mismos requisitos con la visión del sistema pero donde sólo difieren en el lenguaje utilizado. Propone hacer una distinción más profunda en lo conceptual y menos ambigua en cuanto a la terminología, utilizando los nombres “Necesidades del Negocio” para los requisitos de usuarios y “Especificación del Sistema” para los requisitos del sistema, pero además con una visión diferente. Dado que los primeros son derivados del UdeD, se expresan desde el punto de vista del espacio del problema, mientras que los segundos se producen para satisfacer los primeros, entonces se expresan desde el punto de vista del espacio de la solución indicando qué debe hacer el sistema, no cómo lo debe hacer. Ben Kovitz [Kovitz 98] es más categórico en su apreciación: «*Los requisitos definen el problema a ser resuelto por el software; ellos no describen el software que lo resuelve*». Para definir el software, Kovitz utiliza los términos “Especificación” o “Diseño de Interfaz” o “Especificación del programa”, pero con un significado totalmente distinto al de Alexander, pues por un lado, se refiere a la especificación como un concepto y no como la representación de algo (sentencias o diagramas) y por otro, dicha especificación corresponde al espacio de la solución computacional. Esta visión es compartida por Leite [Leite

94]: «*Los requisitos están ligados a lo que se pretende del software, mientras que la especificación procura detallar los mismos bajo una óptica computacional*».

Salvando el uso de "descripciones" que hace Jackson [Jackson 95] para definir requisito y especificación, se puede decir que tanto el término "Requisito" de Kovitz como el término "Necesidades del Negocio" de Alexander coinciden con el término "Requisito" de Jackson, mientras que el término "Especificación" de Jackson es similar al de "Especificación del Sistema" de Alexander y sutilmente diferente en cuanto a la interpretación de Kovitz que lleva la especificación más al terreno del diseño.

Como se puede observar existe una gran variedad de interpretaciones respecto al significado de los términos "requisito" y "especificación". Se debe tener presente que muchas veces en la literatura donde pareciera haber contradicciones, éstas no son conceptuales sino meramente de terminología. Esta tesis se adhiere a la visión de Leite, Jackson y Kovitz respecto al uso del término "requisito". Por eso, se sostiene que «*el conjunto de requisitos debe contemplar las necesidades de los usuarios, las reglas de la organización y las reglas externas próximas al macrosistema*» [Doorn 98].

También existen discrepancias en el uso en español de los términos "requisito" y "requerimiento" como traducción del inglés del vocablo "requirement". Este es sólo un caso más de traducciones obtenidas por transliteraciones más que por comprensión semántica, como ocurre en los casos de "instancia" por "instance" (la verdadera traducción es "ejemplar") o "lecture" por "lectura" (la verdadera traducción es "disertación" o "conferencia") o "trace" por "traza" (la verdadera traducción es "rastro"). Se presentan a continuación definiciones extraídas del Diccionario de la Lengua Española.

> ***requerimiento***. m.
> Acción y efecto de requerir.
> **2.** *Der*. Acto judicial por el que se intima que se haga o se deje de ejecutar una cosa.
> **3.** Aviso, manifestación o pregunta que se hace, generalmente bajo fe notarial, a alguna persona exigiendo o interesando de ella que exprese o declare su actitud o su respuesta
> ***requerir***. *tr*.
> Intimar, avisar o hacer saber una cosa con autoridad pública.
> **2.** Reconocer o examinar el estado en que se halla una cosa.
> **3.** Necesitar.
> **4.** Solicitar, pretender, explicar uno su deseo o pasión amorosa.
> **5.** Inducir, persuadir
> ***requisito***. p.p. irreg. de ***requerir***.
> **2.** Circunstancia o condición necesaria para una cosa
>
> *Diccionario de la Lengua Española*,
> Real Academia Española, XXI edición, Madrid 1992

Se observa que la segunda definición en el glosario de la IEEE [IEEE

Std 610.12-1990] se corresponde con la definición del término ***requisito*** del Diccionario de la Lengua Española, término que se usará en esta tesis adhiriéndose a dicho significado. Por otro lado, se usará el término ***requerimiento***, asociándolo a su primera acepción en el Diccionario de la Lengua Española, para referirse a las necesidades, demandas y deseos, es decir, a los pedidos de un cliente o usuario como entrada al proceso de la IR. Esta definición de requerimiento se asemeja más a la primera definición de la IEEE, mientras que la tercera definición de la IEEE se asemeja al término ***especificación***. Ralph Young en [Young 04] también distingue entre "Requisitos declarados" como las necesidades y expectativas iniciales de los clientes, y los "Requisitos reales" como las necesidades verificadas y validadas para un software.

Si se analizan los términos en inglés "requirement" y "request" se observa que se acoplan mejor a los términos en español "requisito" y "requerimiento" respectivamente.

A continuación se presentan las definiciones extraídas del Diccionario de Inglés Contemporáneo Longman, donde se observa que la palabra en inglés "requisite" está más relacionada con "requirement" y en español con "requisito". Cabe mencionar que la palabra "requirement", generalmente usada en plural, está entre las 2000 palabras más frecuentemente usadas en inglés, tanto escrito como oral. Asimismo en castellano la palabra "requisito" también está entre las 2000 más frecuentemente usadas, siendo tres veces más frecuente su uso en plural que en singular, mientras que la palabra "requerimiento" tiene una frecuencia de uso mucho menor, entre las 5000 más usadas [Almela 05].

requirement
noun
1 something that someone needs or asks for
2 something that must be done because of a law or rule
3 something, especially good examination results, that a college, employer etc says you must have in order to do something

require
transitive verb
1 to need something
2 if you are required to do or have something, a law or rule says you must do it or have it: **be required to do something**
Require is more formal than **need**

request
noun
1 a polite or formal demand for something

requisite
noun, formal
1 something that is needed for a particular purpose -> Prerequisite

Dictionary of Contemporary English: The living Dictionary,
Longman, Pearson Education Limited, England, 2003

Como se puede apreciar en la Figura 2-9, el proceso de la IR tiene diversas entradas, entre ellas los requerimientos, siendo la salida del mismo los requisitos del software, cuya representación son las especificaciones.

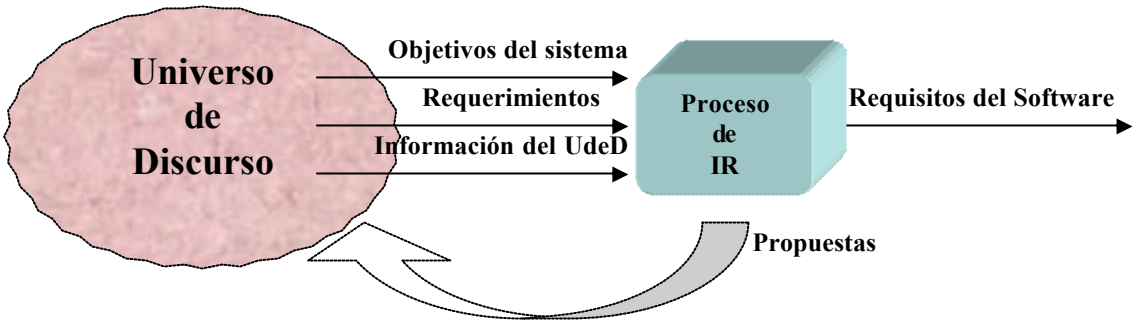


Figura 2-9. Requerimientos versus Requisitos

Con frecuencia se supone que todo pedido, demanda o necesidad presentada por los clientes y usuarios se traslada automáticamente al software. Sin embargo, la práctica real muestra que esta hipótesis suele ser falsa. El par demanda – satisfacción incondicional es una utopía que el desarrollo de software no ha logrado y probablemente nunca alcance.

Las necesidades se gestan bajo una óptica parcial del UdeD, no forzosamente compartidas por todos los involucrados, muchas veces contradictoras con otras necesidades y hasta incluso pueden llegar a no ser genuinas. La satisfacción incondicional puede sustentarse a través de soluciones tecnológicas inexistentes, consideraciones de alcance político o legal, presupuestos ilimitados y cronogramas inviables. Es acá donde hay una clara distinción entre las demandas y lo que efectivamente un sistema de software podrá llevar a cabo, y es parte de las tareas de los ingenieros de requisitos en realizar una transformación de las demandas (requerimientos) en condiciones factibles (requisitos) para el sistema de software a construir.

La información proveniente del UdeD es muy variada en propósito y granularidad, puede ser actual u obsoleta, relevante o fútil, y es el ingeniero de requisitos quien tamiza esa información en el proceso de la IR en estrecha colaboración con los clientes y usuarios. De la Figura 2-9, se observa que las entradas al proceso de la IR son entonces:

- **Objetivos del sistema**: propuestos por los clientes, y representan las características generales o propósito del sistema.
- **Información del UdeD**: proveniente de distintas fuentes de información, muy heterogénea en contenido, en granularidad, en vigencia y en utilidad. Básicamente se puede agrupar en:
  - Actividades actuales del negocio: provenientes principalmente de los usuarios, y corresponden a descripciones de cómo se hacen las tareas.
  - Propiedades o características del UdeD: provenientes de distintas fuentes de información, como usuarios y documentación, y corresponden por ejemplo a estándares de

la organización, regulaciones gubernamentales o políticas de la organización.

- ⇨ **Requerimientos**: provenientes de los usuarios, y representan necesidades, deseos, imposiciones y limitaciones.
- ⇨ **Propuestas**: elaboradas por los ingenieros de requisitos, y representan básicamente ideas para mejorar las actividades del negocio a través del software.

Los requerimientos son manifestados por los usuarios en diferentes niveles de abstracción. Si un usuario da instrucciones específicas sobre algo que el sistema de software deberá hacer, se convierte casi instantáneamente en un requisito del software, y la única consideración posible es determinar si es consistente o coherente con otros requisitos.

Por otro lado, cuando un requerimiento viene planteado en el formato de un problema actual a evitarse o de un deseo futuro poco preciso, el ingeniero de requisitos debe construir uno o más requisitos detallados que bajen ese pedido al nivel de situaciones o características en el UdeD a ser provistos por el sistema de software.

En oposición a los requerimientos, el conocimiento acerca del UdeD se captura en una forma muy detallada. Las actividades del negocio son descripciones del comportamiento del UdeD, descripciones de cómo se realizan. Estas descripciones deben orientarse al porqué se realizan, es decir hay que levantar el nivel de abstracción de dichas descripciones. A cada descripción hay que vincularle su justificación, esto ayuda a que los ingenieros de requisitos realicen propuestas.

Las propuestas del ingeniero de requisitos son una fuente de requisitos habitualmente poco considerada pero muy importante. Estas propuestas se confunden frecuentemente con actividades realizadas durante el Diseño de Software (principalmente cuando no hay una distinción clara entre el equipo de ingenieros de requisitos y el equipo de diseñadores) o con actividades correspondientes a la Reingeniería de los Procesos del Negocio (esto ocurre cuando el ingeniero de requisitos sugiere nuevas formas de hacer las cosas).

Las propuestas son vistas a veces como parte del Diseño de Software por la antelación de esta disciplina frente a la Ingeniería de Requisitos, pero también se debe a que muchas veces se realiza incorrectamente en dicha fase[39]. Esto debe evitarse, dado que muchas veces durante la fase de Diseño no se dispone de toda la información útil para realizar estas propuestas. Por otro lado, las propuestas también pueden ser vistas como parte de la Reingeniería de Procesos, dado que la forma en que el negocio se gestiona efectivamente cambia. Sin embargo, el impacto de estos cambios frente a los objetivos principales del negocio es con frecuencia casi irrelevante, y en ese sentido, esto tiene poco que ver con lo que se denomina tradicionalmente reingeniería del negocio.

[39] Dado que el Diseño de Software tuvo su origen mucho antes que la IR, dicha disciplina pretendió abordar, en sus comienzos, todo tema que no fuera estrictamente relativo a codificación y prueba.

Se entiende entonces por propuestas de los ingenieros de software a aquellas realizadas a los clientes y usuarios sobre cambios en un requerimiento o la incorporación de un nuevo requisito que intenta modificar en algún grado la forma en que los usuarios hacen las cosas o pretenden hacerlas en el futuro. Por supuesto, estas propuestas deben ser explicadas, negociadas y documentadas.

Cabe observar que los requisitos incluidos en el SRS son en realidad requerimientos para el grupo de diseño, considerando que, quien pide algo tiene requerimientos (necesidades) y quien debe realizarlo trata con requisitos (condiciones a cumplir). Por lo que, en esta tesis se prefiere hablar de ingeniería de requisitos, o sea, abarcar desde la captura de las necesidades de los clientes hasta la definición de las condiciones a cumplir por el software. Sin embargo, sería concebible una ingeniería que se dedique solamente a entender lo que el cliente quiere.

2.6.2. Qué versus Cómo

Los requisitos deben especificarse sin expresar alternativas computacionales ni detalles tecnológicos, es decir, mantener las descripciones a nivel del negocio. Muchos investigadores se expresan sobre el dilema entre el qué y el cómo en la definición de requisitos [Faulk 96] [Jackson 95] [Davis 93] [Siddiqi 96] [Dorfman 97] [Kovitz 98].

Se considera como un principio establecido en la comunidad de la Ingeniería del Software que los requisitos deben describir el *qué* debe hacer el sistema prescindiendo de detalles de diseño e implantación, es decir, evitando describir el *cómo* debe hacer las cosas el sistema, dado que esto último corresponde a la fase de diseño del software. Se supone entonces que los requisitos se deben describir en términos del problema y no en términos de la solución, para evitar sobre-restringir el diseño y la implementación. Aunque los requisitos evolucionan, éstos no deberían estar afectados por cambios en el diseño, en la tecnología utilizada o en la implantación del software y ni siquiera por cambios en la gestión del proyecto (cronograma, presupuesto, planes de V&V, recursos humanos).

La división clásica entre el *qué* y el *cómo* es clara y funcional desde el espacio del desarrollador de software, donde qué es lo que hay que hacer proviene de la definición de requisitos y el cómo se va a hacer, lo decide él. Davis [Davis 93] sostiene que la división entre el *qué* y el *cómo* es una cuestión de punto de vista, donde todo artefacto de software representa el *cómo* en una etapa y para una siguiente etapa donde es una entrada representa el *qué* [40].

[40] Esta visión del qué y el cómo es muy similar a lo ya discutido entre requerimiento y requisito (ver sub-sección 2.6.1).

Visión	*Qué* 	*Cómo* 
Cliente	Una condición o capacidad necesaria para el negocio.	Una funcionalidad o capacidad que debe poseer un sistema de software.
Desarrollador	Una funcionalidad o capacidad que debe poseer un sistema de software.	Funciones y atributos de componentes específicos del sistema de software.

Tabla 2-5. Visiones del Qué y del Cómo

Esta división no es tan clara cuando se involucran los clientes y usuarios. Ellos tienen la visión de *qué* quiere el negocio y el *cómo* está dado a través de un sistema de software. O sea, el *cómo* solucionar un problema del negocio, representa el *qué* debe hacer el sistema según el desarrollador. Esta diferencia de visiones se muestra en la Tabla 2-5, mientras que la Tabla 2-6 ejemplifica estas visiones. Pero, por otro lado, ocurre que a menudo los clientes y usuarios quieren intervenir en la definición de cómo el sistema les va a solucionar el problema.

En los párrafos siguientes se adopta la posición del desarrollador, que es la que tradicionalmente se asume como la distinción entre el *qué* y el *cómo*.

Zave y Jackson en [Zave 97] son muy categóricos en su apreciación de los requisitos, para ellos los requisitos no deben contener nada más que información acerca del ambiente: especificar los requisitos mediante sentencias que describan cómo será el ambiente cuando se conecte la máquina (artefacto de software) al ambiente, entonces es innecesario describir la máquina y por ende no se incluyen detalles de implementación.

Visión	*Qué*	*Cómo*
Cliente	Una venta en cuenta corriente a clientes minoristas no debe superar el 20% del crédito acordado.	El sistema debe alertar cuando se emite una factura en cuenta corriente que supere el 20% del crédito acordado a clientes minoristas.
Desarrollador	El sistema debe distinguir entre clientes minoristas y mayoristas. El sistema debe mantener el crédito acordado a clientes minoristas. El sistema debe verificar que el monto total de una factura en cuenta corriente a emitir a un cliente minorista no supere el 20% del crédito acordado.	Tabla Clientes: atributo tipo de cliente (Minorista, Mayorista), atributo crédito acordado (numérico, no nulo para tipo de cliente = Minorista). Módulo Facturación: - calcular monto total de factura - si tipo de cliente = Minorista, verificar que monto total de factura <= 0.20 * crédito acordado

Tabla 2-6. Ejemplificando las visiones del Qué y del Cómo

Aunque muchos autores se adhieren a este principio [Leffingwell 00] [Davis 93] [Jackson 95], y así lo recomienda el estándar IEEE sobre Especificación de Requisitos de Software [IEEE Std 830-1998], debe notarse que la línea entre el qué y el cómo es difusa en muchos casos. No es sencillo establecer qué corresponde al espacio del problema y qué al espacio de la solución. A veces hay imposiciones de diseño provenientes de nuestros clientes y usuarios, las cuales deben respetarse y por lo tanto, deben incluirse como requisitos. Por ejemplo: el pedido de utilizar un determinado lenguaje de programación, o una plataforma específica, o un dado gestor de base de datos, o una compatibilidad de algún tipo con algún sistema existente, o un componente de biblioteca. También puede ocurrir que se adopte una solución en particular acordada tempranamente entre los involucrados, entonces no llegará a una decisión durante el diseño. Hay otras formas más sutiles de detalles de implementación, por ejemplo un requisito puede ser: "*El sistema deberá emitir un reporte con un histograma de las ventas mensuales en un período dado*". Uno se puede plantear varios interrogantes: ¿Es necesario un reporte impreso o basta con verlo en pantalla o exportarlo a otro sistema? ¿Es más conveniente un histograma para comparar información que otro tipo de gráfico? Cabe reflexionar entonces si se están tomando decisiones prematuras

de diseño o si esto es realmente parte de los requisitos. Para el caso de soluciones acordadas (como el ejemplo presentado) se podrán considerar RF que obviamente restringen el diseño.

Es claro que la solución al problema se "diseña" y se acuerda con los clientes y usuarios, pero esto puede o no involucrar aspectos relacionados con la tecnología. ¿Cuál es la línea divisoria donde ese "diseñar la solución" corresponde a la IR o a la fase de Diseño (técnico)? No existe una respuesta tajante, dependerá de diversos factores, tales como el tipo de aplicación, el UdeD, los involucrados y la relación establecida entre ellos, entre otros factores. En tal sentido, Harwell et al. [Harwell 93] sugieren preguntarse "*¿Por qué se necesita el requisito?*" con el fin de determinar si éste es una necesidad genuina en vez de una decisión de diseño adoptada tempranamente. Agregan también que cuando no hay otra forma de implementación más que la expresada en el requisito, entonces éste casi con certeza corresponde a una decisión de diseño y debe ser excluido del documento de requisitos, excepto que su implementación sea dependiente del contexto en el cual dicho requisito será implementado. En una línea similar, Jackson [Jackson 95] observa que cuando uno describe qué hace un sistema, lo que está describiendo realmente es el propósito de ese sistema.

La práctica indica que las actividades de definición de requisitos y de diseño son iterativas e interdependientes [Swartout 82] [Leffingwell 00]: decisiones en IR afectan decisiones de diseño y viceversa. No es posible entonces negar esta realidad y la gestión de requisitos debe lidiar con ella.

2.6.3. Propiedades y Atributos de los requisitos

Las principales propiedades o cualidades que deben presentar los requisitos son:

- *Consistentes:* los requisitos no deben estar en conflicto entre ellos.
- *No ambiguos*: los requisitos deben poder interpretarse de una sola manera.
- *Completos:* los requisitos deben abarcar toda posible entrada al sistema y toda posible respuesta del sistema.
- *Correctos:* los requisitos deben ser realmente necesarios y cumplir con el propósito del sistema.
- *Entendibles:* los requisitos deben poder ser comprendidos por cualquier involucrado con un mínimo de explicación.
- *Realizables:* los requisitos deben poder satisfacerse considerando los recursos disponibles y restricciones, es decir, los requisitos deben ser posibles de llevarse a cabo.
- *Rastreables:* los requisitos deben poder relacionarse bidireccionalmente a las fuentes que les dieron origen, y también a los modelos, documentos y componentes del software.
- *Verificables:* los requisitos deben poder comprobarse a través del software.
- *Abstractos:* los requisitos deben ser independientes de la

implementación.

- ⇨ *Referenciables por importancia y/o estabilidad:* los requisitos deben permitir que se les asignen prioridad y grado de estabilidad que presentan en el tiempo.

Davis et al. [Davis 93b] describen una lista de 24 cualidades que deben poseer los requisitos; la lista antes presentada fue generada a partir de las propiedades propuestas por varios autores, incluyendo el estándar de IEEE [IEEE Std 830-1998] que describe 8 cualidades.

A continuación se describen algunos de los atributos que se asignan a los requisitos para alcanzar una adecuada gestión de los mismos:

- ⇨ *Identificación:* debe indicarse la referencia unívoca al requisito.
- ⇨ *Prioridad:* debe indicarse la importancia relativa que tiene para los clientes y usuarios.
- ⇨ *Criticidad:* debe indicarse la necesidad relativa de implementación, se refiere a si es obligatorio, deseable u opcional.
- ⇨ *Origen:* debe indicarse la fuente de información de donde se capturó el requisito, incluyendo oportunidad en que se detectó.
- ⇨ *Tipo o categoría:* debe indicarse si se trata de un RF o un RNF, o el tipo de RNF al que corresponde, o el tipo según alguna otra clasificación utilizada (ver sub-sección 2.6.4).
- ⇨ *Dependencias:* debe indicarse las relaciones con otros requisitos.
- ⇨ *Estado:* debe indicarse la condición bajo la cual se encuentra el requisito, puede ser por ejemplo propuesto, aprobado, implementado, validado.
- ⇨ *Autor:* debe indicarse el responsable de su identificación y descripción.
- ⇨ *Fundamento:* debe indicarse la razón de la existencia del requisito.
- ⇨ *Versión:* debe llevarse una historia de los cambios (ver sub-sección 2.2.7).
- ⇨ *Fechas de creación y de modificación:* debe indicarse la fecha en que fue identificado y cuándo sufrió modificaciones.
- ⇨ *Vinculación:* debe establecerse los enlaces con los modelos del sistema que le agregan detalle al requisito.
- ⇨ *Factibilidad:* debe indicarse la posibilidad de implementación, ya sea por razones tecnológicas, ambientales, económicas, o políticas. Se califica generalmente como alta, media o baja.
- ⇨ *Riesgo:* debe indicarse una calificación en función de las consecuencias de su implementación. Se califica generalmente como alto, medio o bajo.

Esta lista se basó en los atributos mencionados en el estándar [IEEE Std P1233/D3-1995], en [Davis 99] y en [Kotonya 98]. En general los atributos que se asocian a los requisitos se adaptan según el equipo de desarrolladores, cada aplicación en particular y la herramienta de gestión utilizada. Los atributos asociados a los requisitos le darán contexto a cada requisito, posibilitando su identificación, clasificación o selección, y permitiendo el control mismo del proceso de definición de requisitos [Hull 05]. En [Young 04] además de presentar un ejemplo de atributos, se aconseja no sobrecargar la especificación de requisitos sino definir sólo aquellos atributos que realmente se van a

ingresar y administrar.

Desde el punto de vista de la presentación de los requisitos en formularios, listas o planillas, varios de estos atributos pueden estar factorizados como consecuencia de la taxonomía que se utilice o por ser sus valores idénticos para todos los requisitos (ver sub-sección siguiente).

2.6.4. Taxonomías de requisitos

Existen en la literatura diversas formas de clasificar los requisitos, las que atienden distintos propósitos. Por ejemplo, desde un punto de vista evolutivo, en [Sommerville 02] y en [Kotonya 98] se distinguen a los requisitos estables o duraderos versus los requisitos volátiles, con el fin de facilitar la gestión en los cambios de los mismos. Desde un punto de vista de implementación, se los clasifica en obligatorios versus deseables, asignándoles prioridades para determinar qué desarrollar en la siguiente etapa del proyecto. Oliveira en [Oliveira 98] presenta una taxonomía de requisitos bajo cuatro perspectivas que facilitan la detección de defectos: i) en base al contexto, requisitos externos o internos, ii) en base al nivel de abstracción, requisitos genéricos o detallados, iii) en base a la validación de los clientes, requisitos validados o no, y iv) en base a modelos del sistema, requisitos incluidos (considerados) o excluidos (no considerados). Sawyer & Kotonya [Sawyer 01] presentan una taxonomía de requisitos bajo seis diferentes dimensiones, algunas ya mencionadas, adicionando las siguientes: i) desde la perspectiva del alcance del requisito: si éste afecta al sistema o a un componente, ii) desde la perspectiva de emergencia del requisito: si éste es derivado de otros o emergente directamente de las fuentes y iii) desde la perspectiva de su naturaleza: requisito del producto o requisito del proceso.

En [Young 04] se establece una diferenciación muy importante de los requisitos en general en: requisitos del negocio, requisitos del proceso y requisitos del producto; de manera tal que los requisitos del producto deben atender ineludiblemente a los requisitos del negocio y se separa del software aquellas condiciones que debe cumplir el proceso de desarrollo.

Una clasificación ampliamente difundida corresponde a la de requisitos funcionales (RF) y requisitos no funcionales (RNF). Esta clasificación sirve para simplificar su presentación, hacerlos más legibles, entendibles y rastreables, pero en la práctica esta distinción no siempre resulta tan clara. Aunque, por otro lado, el hecho de distinguirlos obliga a estudiar los RNF que muchas veces son olvidados o desestimados durante el desarrollo del software, y esto ha llevado al fracaso de numerosos proyectos. Los errores y omisiones de RNF son considerados los más caros y más difíciles de corregir [Loucopoulos 95]. Los RNF tienen una gran influencia en la calidad del sistema a ser desarrollado. Mientras que el incumplimiento de un RF degradará al sistema, la no satisfacción de un RNF usualmente provocará la inutilización del mismo [Sommerville 02].

A continuación, se presenta una definición de ambos tipos de requisitos.

Requisito Funcional:
actividades y servicios que debe proveer el sistema. Es decir, los RF expresan el comportamiento esperado del sistema.
En algunos casos, los RF pueden expresar lo que el sistema no debe hacer [Sommerville 02].

Requisito No Funcional:
características y atributos del sistema, así como también cualquier restricción que pueda limitar una solución. Es decir, los RNF expresan restricciones y cualidades que el sistema debe cumplir o poseer.
En algunos casos, los RNF pueden referirse a restricciones al proceso de desarrollo y también a restricciones derivadas del UdeD [Sommerville 02].

En resumen, se puede decir que los RF establecen qué debe hacer el sistema mientras que los RNF restringen cómo el sistema debe cumplir con el qué debe hacer. Es por ello que con frecuencia muchos RNF se vinculan a RF [Chung 00] [Sommerville 02]. Cabe aclarar que en la definición de Sommerville se están incluyendo como RNF restricciones al proceso de desarrollo, que en realidad deben considerarse como tales y no como restricciones al software.

Asimismo los RNF también han sido objetos de variadas clasificaciones. La clasificación de Yeh & Ng [Yeh 90] tiene un sesgo hacia la gestión del proceso de desarrollo, agrupando 3 grandes categorías: i) restricciones del sistema (eficiencia, confiabilidad, seguridad, etc.), ii) restricciones de desarrollo, evolución y mantenimiento (estrategias de desarrollo, escala de esfuerzo, criterios de aceptación, etc.), y iii) restricciones económicas del desarrollo (costo marginal, costo de iteración y costo del software). Una clasificación muy difundida, por incluir muchos de los tipos definidos en otras, es la de Sommerville [Sommerville 02] que también distingue 3 grandes categorías: i) requisitos del producto (usabilidad, eficiencia, confiabilidad y portabilidad), ii) requisitos del proceso (entrega, implementación y estándares), y iii) requisitos externos (interoperabilidad, éticos y legales). También los estándares internacionales ofrecen variadas clasificaciones de RNF, entre ellos: [IEEE Std 830-1998] estipula 9 tipos de RNF, [ESA PSS-05-0] define 13 tipos de RNF, [ISO 9126:2001] define 6 tipos de RNF.

En general la literatura atribuye la falencia en el tratamiento de RNF en los métodos de desarrollo de software a la dificultad que esto representa [Loucopoulos 95] [Kotonya 98]. Los RNF poseen una naturaleza abstracta e intangible en comparación con los RF [Breitman 99], y esto hace que sean más difíciles de especificar o documentar formalmente. Los RNF frecuentemente tienden a entrar en conflicto entre sí [Loucopoulos 95] [Cysneiros 04], lo cual debe poder identificarse y resolverse. Pero, por otro lado, los RNF son potencialmente sinérgicos, el cumplimiento de uno o más RNF puede incidir positivamente en el cumplimiento de otros RNF [Loucopoulos 95]. La relevancia de un RNF es dependiente de la naturaleza del sistema [Loucopoulos 95], y por ende su descripción es relativa al dominio particular. Otra dificultad que a veces presentan los RNF es que son difíciles de verificar, deberían poder expresarse

cuantitativamente de manera que puedan probarse en forma objetiva [Sommerville 02]. Algunos RNF están relacionados con soluciones de diseño que se desconocen durante el proceso de definición de requisitos [Kotonya 98]. Dado que los RNF tienden a vincularse con uno o más RF, si se describen por separado los RNF de los RF en el SRS, entonces se dificulta encontrar la relación existente entre ellos, pero si se los expresa juntos se dificulta el establecer cuáles son consideraciones funcionales y cuáles no [Kotonya 98].

Recién en los últimos años se han propuesto métodos para considerar adecuadamente los RNF [Mylopoulos 92] [Kotonya 93] [Boehm 96] [Cysneiros 04] [Kerkow 05]. A continuación se describen brevemente algunos de estos métodos.

El enfoque propuesto por Kotonya & Sommerville [Kotonya 93] [Kotonya 96] denominado VORD se basa en puntos de vista de los clientes, de la organización, de la ingeniería del sistema y externos, e integra RF con RNF. Se construye una jerarquía de especialización de puntos de vista a los que se les asocia RF y RNF, los cuales por ende también se especializan. Los RNF a su vez son vinculados a los RF que afectan.

Boehm & In [Boehm 96] propone el uso de una base de conocimiento donde los RNF, que denomina requisitos de calidad, son priorizados según el punto de vista de los clientes y usuarios. La propuesta se basa en la identificación de conflictos entre los RNF deseados y establecer un balance para la satisfacción de los mismos.

Mylopoulos, Chung & Nixon [Mylopoulos 92] han propuesto un método basado en modelos orientados a objetos denominado NFR Framework. Este método considera a los RNF como objetivos que pueden entrar en conflicto entre ellos y deben representarse como “softgoals” a ser satisfechos, los cuales se descomponen en sub-objetivos hasta llegar a sub-objetivos satisfacibles (denominados operacionalizaciones[41]). La representación utilizada para cada RNF es un grafo basado en árboles de decisión AND/OR.

El enfoque de Cysneiros & Yu [Cysnerios 04] se basa en el método NFR Framework [Mylopoulos 92] y para la elicitación de RNF usa una estrategia basada en la construcción del LEL [Leite 93] (ver capítulo 4). En el LEL se agregan RNF deseados por los clientes y usuarios, y algunas de sus operacionalizaciones. Para facilitar la identificación de estos RNF y sus operacionalizaciones, se usan catálogos de RNF. Se cuenta para ello con una base de conocimiento que tiene registrados estos catálogos, los cuales pueden actualizarse o pueden crearse nuevos. Luego, los RNF identificados se representan en grafos usando el NFR Framework con adaptaciones y se buscan posibles interdependencias en el conjunto de grafos.

En la práctica es muchas veces difícil distinguir si un requisito es funcional o no funcional; muchas veces esto depende del grado de detalle con que se describe al requisito, si se lo describe en forma más abstracta o general

[41] Operacionalización: son especificaciones funcionales que surgen por la necesidad de cumplir con cierto RNF [Cysnerios 04].

fácilmente podrá considerarse como un RNF pero al describirlo con más detalle podrá parecer un RF [Kotonya 98]. Este es el caso en que un RNF es descompuesto en varios RF más específicos (operacionalizaciones). Otras veces no es fácil establecer a qué categoría pertenece un determinado RNF. Como resalta Leffingwell & Widrig en [Leffingwell 00], la clasificación de los requisitos no es un tema tan importante en sí mismo que retrase un proyecto por dificultades en la inclusión de un determinado requisito en tal o cual categoría, la clasificación adoptada debe contribuir a promover la elicitación y modelado de todos los requisitos, siendo una guía para evitar principalmente omisiones.

En todas las actividades de transformación del *qué* en *cómo* (ver sub-sección 2.6.2) tanto durante el proceso de definición de requisitos (punto de vista del cliente) como en el diseño e implementación del software (punto de vista del desarrollador) se procede a transformar gran parte de los RNF en RF. Los RNF están entonces presentes fundamentalmente en el *qué*. En consecuencia podría decirse que un buen diseño de un sistema de software debiera carecer de RNF del producto y externos residuales, ya que todo RNF existente con anterioridad se debiera haber transformado en una funcionalidad del sistema o característica de su arquitectura.

2.6.5. Documento de Definición de Requisitos

Los requisitos son generalmente documentados con una narrativa explícita para comunicarlos a los involucrados en el proyecto. Este es el objetivo central de la generación de este documento [Loucopoulos 95], el cual también sirve como contrato entre los clientes y el equipo de desarrollo sobre lo que va a proveer el nuevo sistema. Con este documento, los clientes tienen un acuerdo de lo que los desarrolladores deben producir. Para los desarrolladores, les define el alcance del proyecto y pueden prevenir que los usuarios agreguen adicionales al alcance sin impactar el cronograma, el presupuesto y los recursos asignados.

Además, el SRS puede ser usado para registrar y gestionar cómo los requisitos son satisfechos y qué fragmento del sistema fue originado en un cierto requisito (requirements allocation), como un ancla para pre y post rastreo de los requisitos, como ancla para la gestión de cambios en los requisitos, como herramienta de validación de los requisitos con los clientes, como puente de comunicación entre el equipo de analistas y el equipo de diseñadores, como instrumento de los testeadores para validar el sistema de software.

En diferentes organizaciones y en la literatura misma, este documento recibe variados nombres, desde documento de requisitos (el nombre más informal) hasta especificación de requisitos de software (SRS) (el nombre más formal), y se genera con diversas estructuras [Kotonya 98] [Leffingwell 00] [Fairley 96], aunque existen estándares internacionales que especifican su estructura, tales como [IEEE Std 830-1998] (ver Figura 2-10), [IEEE Std P1233/D3-1995], [DOD 5000.2-R], [ESA PSS-05-0], [NCC 87] entre otros. Un estándar reciente para el documento de definición de requisitos es [IEEE Std

1362-1998], bajo el nombre de "Concept of Operations" (ConOps)[42] que describe los requisitos de un sistema propuesto desde el punto de vista del usuario.

Este documento es el producto final del proceso de la IR; en algunos proyectos pequeños y medianos puede ser muy informal, e incluso puede no existir. En general, se escribe en lenguaje natural para servir como medio de comunicación entre los participantes, pero en algunos casos se lo escribe con una notación formal o semi-formal para describir los requisitos en forma más precisa y concisa [Sawyer 01]. Se requiere entrenamiento especial para generar un SRS en LN que sea lo más completo posible, consistente, no ambiguo, legible y además producirlo en un lapso de tiempo razonable [Kovitz 02].

Un documento de definición de requisitos debe describir:

- ⇨ funciones y servicios que el sistema proveerá (RF),
- ⇨ propiedades que debe poseer el sistema (RNF),
- ⇨ restricciones que limitan el proceso de desarrollo del sistema y restricciones bajo las cuales el sistema debe operar (RNF),
- ⇨ información acerca de otros sistemas con los que el sistema debe integrarse, e
- ⇨ información sobre el dominio de la aplicación.

Como se observa en la Figura 2-10, la estructura que sugiere la IEEE es una guía general de lo que debería contener este documento, dando pautas generales de contenido de cada sección. La sección 1 describe los objetivos y alcance del sistema, e incluye un glosario de términos utilizados en el documento. La sección 2 describe la funcionalidad general del sistema y los factores que afectarán al sistema y al proceso de desarrollo. La sección 3 describe en detalle los requisitos del software. Los apéndices pueden ser parte del documento o estar separados, ellos pueden ser modelos del sistema que se adosan para clarificar las descripciones de los requisitos, descripciones de hardware, descripciones de base de datos a utilizar, componentes de software a ser reusados, etc.

[42] Este documento denominado Concept of Operations ha ido evolucionando desde su primera formulación en 1980 [Lano 80].

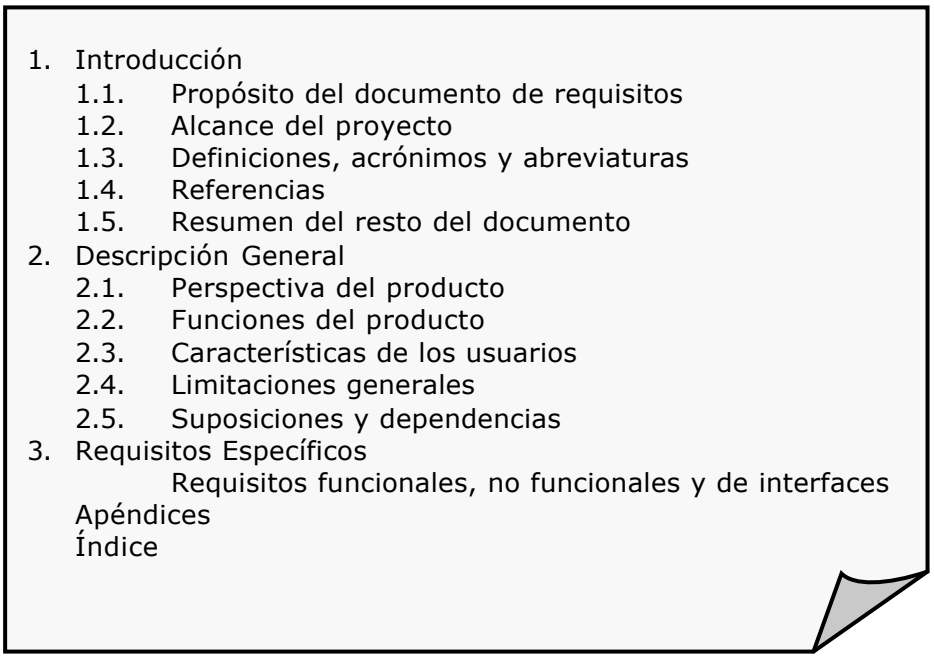


Figura 2-10. Estructura de SRS según [IEEE Std 830-1998]

Se desprende de la Figura 2-10 que la sección 3 del documento no tiene una apertura en sub-secciones, el detalle de cómo organizar y describir los requisitos específicos dependerá de la organización, de la estrategia de desarrollo a utilizar y del tipo de software a desarrollar, pero sí ofrece ocho patrones guías alternativos de cómo organizar la sección.

Cabe aclarar que no siempre se crea el SRS. Los requisitos pueden quedar empotrados en los modelos construidos, o definirse con alguna plantilla donde se incluyen ciertos atributos [IEEE Std P1233/D3-1995] [Davis 99] [Kotonya 98] [Sawyer 01] [Whitten 03] [Young 04].

2.7. Uso de Glosarios en el desarrollo de Software

2.7.1. Introducción a diccionarios y glosarios

Cuando se le preguntó a K'ung-fu-tzu (Confucio) [Eliot 01] cómo gobernar, él dijo «*poniendo los nombres correctos*» y agregó «*Si los nombres no son correctos, se emplean mal las palabras. Cuando se emplean mal las palabras, los asuntos salen mal*». Los dichos de Confucio remarcan la importancia de las palabras, sus nombres y cómo éstas son usadas para lograr el entendimiento del mensaje. Por lo tanto, los diccionarios y glosarios son la mejor forma que el hombre ha encontrado para obtener la comprensión de las palabras de manera tal de usarlas apropiadamente. La palabra “diccionario” fue usada por primera vez por Henry Cockeran en 1623 pero los primeros diccionarios que se conocen datan del siglo VII AC y contiene la información más relevante sobre la cultura mesopotámica [Encarta 04]. Los primeros diccionarios fueron catálogos de palabras y frases inusuales, difíciles o confusas, dado que se consideraba que el vocabulario común no requería de

una explicación o definición. El glosario más viejo conocido proviene del siglo II DC y contiene palabras técnicas griegas usadas por Hipócritas. Siglos más tarde recién se creó un catálogo de todas las palabras de un idioma: el árabe. Es así, que el origen de glosarios y diccionarios fue dar definiciones de palabras y frases de un dominio particular, posteriormente se extendieron a un idioma entero: los diccionarios léxicos. Pero hoy en día, existen diferentes tipos de diccionarios que cubren diferentes necesidades, como los diccionarios de sinónimos y antónimos, los diccionarios de uso idiomático, los diccionarios etimológicos, los diccionarios enciclopédicos, los diccionarios bilingües, los glosarios en los libros de texto, los diccionarios de ideologías, los diccionarios de jergas, los diccionarios de neologismos, entre otros.

Los diccionarios y glosarios comparten muchas características y podrían ser no discernibles a primera vista. Sin embargo, ellos difieren en varios aspectos, algunos de ellos de una manera muy sutil. Las diferencias principales residen en su objetivo, alcance y tamaño. Los diccionarios se estructuran para dar la mayor cantidad posible de significados de cada vocablo, mientras que los glosarios se crean en general para precisar un significando de una palabra en un contexto definido.

Los glosarios no sólo se usan como un producto final para entender el significado de un término a través de su definición o de información acerca de ellos, sino también como productos intermedios o auxiliares en muchos tipos de procesos que requieren la manipulación de texto. Por ejemplo, las tecnologías de traducción, ya sean automáticas o asistidas por computadora [Craciunescu 04] [Hutchins 92] [Hutchins 98] [Trujillo 99], hacen uso de diversas herramientas como los diccionarios bilingües electrónicos, glosarios, bases de datos de terminología y memorias de traducción. En este campo, varias metodologías de traducción de texto técnico implementan la creación de un glosario como una actividad inicial, la cual es responsable de extraer la terminología importante de los documentos fuente; esto se hace porque una buena traducción se apoya en el significado de algunas palabras claves específicas en su contexto técnico. Otro ejemplo es la localización de software [Esselink 00], la cual generalmente involucra, como parte del proceso, la creación y mantenimiento de dos tipos de glosarios. Uno es un glosario general con términos específicos del producto, y el otro es un glosario de interfaz de usuario con todas las opciones que aparecen en los menúes, comandos, cajas de diálogo y mensajes de error. Aquí, los glosarios contienen una lista de términos con su traducción al idioma del mercado destino dónde el producto de software va a ser usado.

Los glosarios se han usado en la ingeniería de software con diferentes propósitos, tales como los diccionarios de datos en los primeros libros sobre bases de datos [Codd 82] para documentar las entidades, atributos, relaciones, tipos y servicios de una base de datos. Así, proporcionan una comprensión común de todos los nombres utilizados en el sistema para el equipo de desarrolladores y luego para el equipo de mantenimiento. Pero, los diccionarios de datos también son un componente importante del análisis estructurado, registrando los datos, almacenamientos de datos y el detalle de los procesos [Senn 89] [Gane 79], aunque autores como Gane & Sarson sugieren que el

nombre real para ellos debería ser “guía del proyecto” en lugar de diccionario de datos. Estos diccionarios de datos se crean durante el análisis pero también se usan durante el diseño del sistema. Ellos satisfacen cinco objetivos: manejar los detalles, comunicar significados comunes, documentar las características del sistema, ayudar el análisis de detalles y cambios, y localizar los errores y omisiones del sistema.

Los diccionarios y glosarios relacionados con el software eran creados para uso interno, es decir, como una fuente de referencia para desarrolladores de software; no tenían el propósito de involucrar el lenguaje del dominio de la aplicación, por consiguiente, podían estar reflejando errores provocados por la incomprensión del UdeD.

Los glosarios también son un tópico muy común en los sitios web, proporcionando definiciones y navegación extra. La mayoría de estos glosarios son sólo de lectura, sin embargo, en algunos sitios los usuarios pueden agregar, borrar y modificar términos, insertar comentarios e ilustraciones, e incluso crear nuevos glosarios asociados a un sitio o documento web. Es decir, los usuarios de estos sitios pueden proporcionar su propia comprensión de los términos.

Desde otra perspectiva, muchos estándares de codificación para lenguajes de programación, al especificar las convenciones de nombres, sugieren el uso de los términos que ya existen en el dominio de la aplicación en lugar de crear nuevos nombres. Por lo tanto, los nombres deben llegar a la fase de codificación provenientes de actividades previas del desarrollo y entonces un glosario conteniendo el vocabulario del dominio de la aplicación sería un medio simple por transmitir dichos nombres, elicitados durante la definición de requisitos; pero esto rara vez ocurre.

Las mismas necesidades que han dado origen a glosarios en diferentes contextos demandan ahora su uso en el campo de la ingeniería de requisitos. Además, la necesidad de un glosario en la IR parece algo natural para establecer una comprensión mutua entre los involucrados, e incluso dicho glosario podría usarse a lo largo del ciclo de desarrollo del software para mantener fluida la comunicación establecida.

2.7.2. Estrategias en la IR que usan glosarios

Crear un glosario en la IR no sólo tiene un propósito comunicativo, también da información a los ingenieros y los hace participar en una realidad diferente: el dominio del problema. A lo largo del proceso de desarrollo de software, este glosario puede servir como un medio colectivo de comprensión.

Muchos autores proponen el uso de glosarios durante la fase de definición de requisitos, pero en la mayoría de los casos su uso se restringe a un rol secundario de referencia y búsqueda. Tal es el caso de Constantine [Constantine 98] que propone en su proceso Joint Essential Modelling la construcción de un glosario como un producto optativo. A continuación se

presenta un resumen de propuestas de investigadores y de productos comerciales que emplean glosarios en la IR.

El enfoque propuesto por Rolland & Ben Achour [Rolland 98c] incluye el uso de un glosario de términos asociados a una familia de casos de uso. Se genera una lista de términos clasificados en Agentes, Recursos y Acciones (según el ejemplo presentado en dicho artículo). Tratan las ambigüedades e inconsistencias del uso de LN en casos de uso y escenarios usando dicho glosario.

Oberg et al. [Oberg 98] propone el uso de un glosario con los términos comunes a todo participante para ayudar a la comunicación y el entendimiento. Este glosario es refinado durante las actividades de desarrollo y se usa para describir las características del sistema y el modelo de casos de uso. Acuerdan las definiciones del vocabulario con los usuarios, como una especie de validación.

Alspaugh et al. [Alspaugh 99] crean un glosario por cada escenario mientras escriben los escenarios. Estos glosarios les permiten obtener descripciones consistentes del sistema, ayudando a la estrategia de comparación de escenarios. En un caso donde usaron los escenarios para definir requisitos, tuvieron la necesidad de incluir términos generales y términos específicos del dominio.

Regnell et al. proponen en el proceso UORE [Regnell 99b] el uso de un diccionario de datos para unificar la terminología utilizada en las descripciones de actores y casos de uso. Los términos del diccionario son objetos del dominio del problema y sus atributos. La fase de análisis de UORE consiste en identificar actores y casos de uso, y en unificar la terminología en una modalidad iterativa.

Whitenack [Whitenack 94] en el lenguaje de patrones RAPPeL acentúa la creación y mantenimiento de un glosario de términos durante la definición de requisitos, dado que los términos deben entenderse antes de entregar una especificación de requisitos correcta. El glosario no sólo se usa para anclar los términos sino también como una posible fuente para encontrar y definir los objetos del dominio de la aplicación. Whitenack señala que muy a menudo los términos tienen diferentes significados para diferentes personas dentro de la misma organización y es por ello que considera esencial comenzar creando el glosario de términos.

Kovitz [Kovitz 98] sugiere el uso de un glosario que brinde soporte a los documentos de requisitos. Este glosario contendrá principalmente términos del UdeD, cuyas definiciones se irán refinando a lo largo del proyecto. Presenta guías de estilo y contenido para crear el glosario, recomendando que en la definición de cada término se identifique el uso de otros términos del glosario. Al igual que Whitenack y esta tesis, Kovitz señala el problema de los homónimos, y considera que deben conservarse todos los significados diferentes, numerando cada significado y definiendo cada uno en párrafos separados dentro de la misma entrada. También sugiere incluir información

adicional a la definición de un término para clarificarlo.

El Proceso Unificado [Jacobson 99] sugiere la construcción de un glosario con términos comunes e importantes a utilizar en la descripción del sistema. Este glosario tiende a estar más centrado en el sistema a construir que en el contexto del sistema aunque se lo construye como complemento del modelo del negocio.

Díaz et al. [Díaz 04] proponen la construcción de una ontología de términos significativos durante el proceso de generación automática de diagramas de secuencia a partir de casos del uso. Esta ontología se crea extrayendo de cada caso de uso términos que contienen información sobre el dominio. Los términos se definen por su significado y su significante. El significado incluye la denotación y la connotación del término mientras que el significante se usa para reconocer los sinónimos. Consideran que cada término tiene un único significado, vinculándole un rol semántico en el contexto de cada sentencia en el caso de uso, luego un término puede tener más de un rol semántico en el contexto de uno o más casos de uso. Utilizan el principio de máxima circularidad (ver capítulo 4) para describir cada término.

El producto comercial Volere Requirements Specification Template [Robertson 01] [Robertson 99] incluye la creación de un diccionario con las definiciones de todos los nombres usados en la especificación de requisitos, reflejando la terminología de uso actual, a menos que los nombres sean demasiado ambiguos o contengan diferentes significados, y los sinónimos son rechazados. Este diccionario podría usarse para especificar requisitos de los datos y algunos requisitos no funcionales. Se crea durante la definición de requisitos, se valida con los usuarios y se mantiene durante todo el proyecto.

El estándar de la IEEE para la Especificación de Requisitos de Software [IEEE Std 830-1998] sugiere incluir, como parte del SRS o en un anexo, un glosario con los términos utilizados en dicho documento, principalmente para evitar la ambigüedad de ciertos términos y promover la consistencia interna del documento.

El estudio de Weidenhaupt et al. [Weidenhaupt 98] muestra que cuatro proyectos industriales de un total de quince usaron glosarios dentro de enfoques basados en escenarios. Estos glosarios se emplearon principalmente para alcanzar una comprensión común de los términos entre los involucrados y para describir escenarios, también sirvieron para ayudar a los nuevos participantes a familiarizarse con la terminología del proyecto y para buscar escenarios usando una relación bidireccional entre ellos y el glosario.

Como Ben Achour et al. [Ben Achour 99] declaran, el LN proporciona muchas palabras o frases que introducen ambigüedad. A través de un estudio empírico, obtuvieron que el 50% de los casos de uso contenían errores de terminología. Redujeron la proporción de errores de terminología proporcionado guías de estilo para describir los casos de uso. La terminología es una de las cuatro esquinas oscuras que citan Zave & Jackson [Zave 97]: *«Toda la terminología usada en IR deberá estar fundada en la realidad del ambiente*

para el cual se construirá una máquina». Se refieren al problema de usar los términos libremente sin darse cuenta de su exacto significado hasta más adelante. Dado que no es común generar explicaciones de los términos y esto es apenas mencionado por los métodos de requisitos, Zave & Jackson sugieren, como una buena práctica, describir clara y precisamente cada término primitivo del dominio, que denominan "designación".

Revisando la mayoría de los métodos de desarrollo de software [Whitten 03] [Jacobson 99] [RUP 98] [MSF v.3] [METRICA v.3], se podría concluir acerca de la importancia de tener a mano un glosario o diccionario de datos, cubriendo múltiples necesidades, tales como anclar la terminología común entre los desarrolladores, comprender el vocabulario de los clientes y usuarios, servir como un cúmulo de información de fácil búsqueda, entre otros usos. Pero, en general, pocas estrategias dan pautas o principios para crear estos glosarios o diccionarios.

Una buena contribución en esta área sería describir un proceso sólido de creación de glosarios, el cual podría implementarse con pocos cambios en cualquier fase del desarrollo de software, cubriendo diferentes propósitos.

2.8. Escenarios y Casos de Uso

2.8.1. Definición de Escenario y Caso de Uso

La palabra escenario[43], definida como "*a situation that could possible happen*" en el Diccionario de Inglés Contemporáneo Longman y como "*lugar donde se desarrolla cada escena de la película, en el cine*" en el Diccionario de la Lengua Española, ha sido usado durante mucho tiempo en varias áreas de diferentes disciplinas (la estrategia militar, toma de decisiones, marketing, economía). El área de sistema de información comenzó usando los escenarios en la interfaz hombre-máquina y posteriormente en la ingeniería de software. Jarke, Bui & Carroll [Jarke 98b] han proporcionado una visión más amplia de escenarios, tanto desde el punto de vista de los sistemas de información como de una visión de gerenciamiento (teoría de decisión). Señalan que la efectividad del uso de escenarios en varias disciplinas se debe fundamentalmente a la capacidad que poseen los escenarios de estimular el pensamiento. Los escenarios proporcionan una visión de las tareas como situaciones junto con una forma eficaz de comunicación entre los involucrados en el problema bajo estudio.

En el marco de esta tesis, los escenarios describen situaciones que ocurren en el UdeD [Zorman 95] y, teniendo en cuenta aspectos de uso, los escenarios permiten: conocer el problema, unificar criterios, ganar el compromiso de los clientes y usuarios, organizar los detalles involucrados, entrenar a nuevos participantes [Carroll 95], y proveer un ancla para la trazabilidad de los requisitos [Leite 97]. El uso de escenarios, como una técnica

[43] En la comunidad hispanoparlante se ha preservado el uso de la palabra "escenario" frente a "escena" por ser la traducción directa de "scenario", a pesar que el significado del vocablo "escena" es más próximo al de "scenario".

para comprender el problema a ser resuelto usando un sistema de software, ha sido recomendado por varios autores [Potts 95] [Booch 91] [Jacobson 92] [Zorman 95], y sus propuestas han sido muy importantes para extender el uso de escenarios en la práctica real. Sin embargo, un análisis detallado de las recomendaciones dadas en la literatura muestra algún grado de dispersión y contradicciones en el uso de escenarios. El trabajo realizado por el proyecto CREWS confirma esta observación [Rolland 98b] [Weidenhaupt 98].

La falta de precisión sobre cuándo y cómo deben usarse los escenarios, se ha difundido en los ingenieros que están usando estas técnicas en la práctica real. Así, la mayoría de los desarrolladores ven la creación de escenarios más como una destreza que como una tarea de ingeniería. Los estudios acerca del uso de escenarios en los proyectos industriales [Weidenhaupt 98] han demostrado este hecho claramente y observan la necesidad de definiciones más detalladas en la construcción de escenarios con el fin de incrementar su uso en la práctica real. Esto está en completo acuerdo con lo expresado por Rolland et al. [Rolland 98c] y por Sutcliffe [Sutcliffe 97], quienes piensan que hay poca comprensión sobre el uso y producción de escenarios. A pesar de estas observaciones, un estudio reciente sobre la práctica de la IR [Neill 03] comprobó que más del 50% de las organizaciones utilizaba escenarios y casos de uso.

Por otro lado, existen varios estilos[44] para construir escenarios, tales como narrativa textual, storyboards, video mock-ups, prototipos escritos, diagramas de secuencia, redes de Petri, entre otros [Potts 94] [Booch 91] [Jacobson 92] [Carroll 95] [Zorman 95], siendo los casos de uso uno de los tantos estilos de escenarios. Pero tanto Jacobson, el promotor de los casos de uso en el área de informática, en su método OOSE [Jacobson 92], como posteriormente el Proceso Unificado [Jacobson 99], hacen otra distinción entre los términos “caso de uso” y “escenario”. A continuación se brindan ambas definiciones.

> “Un ***caso de uso*** especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo y que producen un resultado observable de valor para un actor concreto”
>
> “Un ***escenario*** es una secuencia especifica de acciones que ilustran un comportamiento”
>
> [Jacobson 99] [Cockburn 00]

En el Proceso Unificado [Jacobson 99] los casos de uso representan principalmente una forma de usar el sistema, capturando RF y algunos RNF, aunque en el Flujo de Modelado del Negocio representan los procesos de negocio de la organización. Mientras que, otros autores [Potts 94] [Kyng 95] [Zorman 95] [Carroll 95b] [Scalzo 95] consideran que los escenarios permiten modelar, además de la funcionalidad del sistema, el comportamiento del

[44] La vista Forma en el marco de trabajo de CREWS [Rolland 98b].

entorno en el cual éste operará. Esta última postura brinda una visión más acabada del problema a resolver, pues mira aspectos sociales del negocio y permite encarar el impacto de incorporar el sistema en el UdeD.

En el marco de esta tesis, se adopta una visión de escenarios en la IR, por lo tanto, sus objetivos principales son capturar requisitos, servir de medio de comunicación entre los involucrados y proveer un soporte para la trazabilidad de los requisitos. Sin embargo, cabe resaltar que la técnica de Escenarios puede ser utilizada a lo largo del ciclo de vida del software con diferentes propósitos [Alexander 04] [Firesmith 94] [Carroll 95] [Rolland 98b], como ser para tomar decisiones de diseño, para generar casos de prueba, para diseñar el entrenamiento y documentación de usuarios.

2.8.2. Estrategias basadas en escenarios en la IR

En la literatura sobre escenarios, existen autores que proponen un proceso de construcción de escenarios basado en un enfoque top-down, es decir, partir de uno o más escenarios de alto nivel de detalle y construir los siguientes escenarios en una forma de refinamiento por pasos. Por otro lado, existen otros autores que consideran que los escenarios deberían construirse desde lo particular a lo general. Algunos de ellos no recomiendan un enfoque específico pero generalmente existe un principio subyacente top-down o bottom-up. Es fácil encontrar dicho enfoque observando cuidadosamente las heurísticas de construcción que proponen.

Por ello, esta sub-sección muestra la distinción entre estrategias top-down y bottom-up, citando algunas de las heurísticas de construcción de escenarios propuestas en la literatura. Esta revisión es importante pues la estrategia que se presenta en esta tesis sigue un enfoque middle-out, basada en la propia experiencia de haber usado previamente enfoques top-down y bottom-up.

Aunque no está claramente establecido, Booch [Booch 94] parece adherirse al enfoque top-down dado que: *«La aplicación más compleja puede caracterizarse en términos de unas pocas docenas de escenarios primarios. Los Escenarios Primarios modelan el problema central. Un Escenario Primario representa alguna función fundamental del sistema. Los Escenarios Secundarios representan alguna variación en el tema de un Escenario Primario. El comportamiento deseado íntegro de un sistema de software puede ser capturado a través de una red de escenarios, de la misma forma que un storyboard hace a una película»*.

Los escenarios primarios de Booch pueden verse como los escenarios relevantes de Firesmith [Firesmith 94], los cuales se interconectan por un diagrama de ciclo de vida de escenarios: *«El diagrama de ciclo de vida de escenarios se utiliza para documentar las interacciones válidas entre los escenarios de mayor nivel de un sistema o ensamblado." Y "Como los escenarios son abstracciones funcionales, existe a menudo una fuerte tendencia a descomponerlos funcionalmente»*.

La propuesta de Sutcliffe [Sutcliffe 97b] también puede incluirse en el enfoque top-down teniendo en cuenta las heurísticas: *«1. Captura inicial de requisitos y familiarización con el dominio. Un escenario fundamental se desarrolla basado en el análisis preliminar del dominio, 2. Especificación y desarrollo de un demostrador de conceptos (prototipo temprano). Diagramas de justificación de diseño se usan para explicar opciones de diseño en los puntos claves, 3. Análisis de requisitos - sesión de validación para criticar el demostrador de conceptos»*.

El proceso de Formalización de Escenarios usando gramáticas regulares en un árbol de escenarios descripto por Hsia et al. [Hsia 94] es también un ejemplo de enfoque top-down.

Por otro lado, la propuesta de Dano et al. [Dano 97] está cercana al enfoque bottom-up, dado que proponen *«recolectar y describir casos de uso basados en una notación tabular»* y luego *«crear redes de Petri para cada caso de uso para brindar el formalismo necesario al analista" y finalmente "establecer vínculos formales entre casos de uso para obtener una descripción global de la aplicación»*. En el artículo de Dano los términos escenarios y casos de uso son usados como sinónimos.

Al igual que Dano et al., Robertson [Robertson 95] también identifica la necesidad de establecer vínculos entre los escenarios: *«Dado que un solo escenario da información parcial, el análisis basado en escenarios involucra la observación de un conjunto de escenarios»* y *«La técnica sistemática de preguntas-respuestas ayuda a crear un puente entre los eventos de un escenario específico y las situaciones generales que posibilitan, causan o explican eventos»*.

El Modelo de Ciclo de Indagación de Potts et al. [Potts 94] puede incluirse también en el enfoque bottom-up teniendo en cuenta: *«Los escenarios se representan en dos niveles de detalle: 1. Episodios o fases que son secuencias de acciones de granularidad fina, 2. Familias de Escenarios usando relaciones de uso de Jacobson»*.

Es además útil observar la construcción de casos de uso, donde enfoques top-down pueden encontrarse, tales como en las heurísticas de [Wirfs-Brock 95] considerando los pasos recomendados: *«1. Determinar el sistema de software (límites, alcance, identificar actores, identificar interfaces con el sistema, desarrollar el primer corte para el particionamiento en subsistemas), 2. Estereotipar los actores (activos o pasivos, roles), 3. Determinar los casos de uso del sistema (descomponer el sistema en porciones discretas significativas tanto para la gente de negocio y los desarrolladores), 4. Construir conversaciones (secuencia de interacciones entre los actores y el sistema para cada caso de uso)»*.

También la propuesta de Oberg et al. [Oberg 98] corresponde a un enfoque top-down, dado que los pasos propuestos involucran: *«1. Análisis del Problema»*, *«2. Comprender las necesidades de los involucrados»*, que incluye

«Encontrar los actores y los casos de uso» y *«Delinear el modelo de casos de uso»*, *«3. Definir el sistema»*, que incluye las tareas: *«Refinar el modelo de casos de uso» y "Describir los casos de uso»*, continúa con: *«4. Administrar el alcance del proyecto»*, que involucra *«Dar prioridades a los casos de uso»* y *«Modelar una vista de casos de uso de la arquitectura del sistema»*, sigue *«5. Refinar la definición del sistema»* que incluye: *«Detallar los casos de uso»*, y finalmente *«6. Administrar cambios en los requisitos»*.

La propuesta de Schneider & Winters [Schneider 98] incluye cuatro fases primarias siguiendo un enfoque top-down: *«1. Fase de concepción: identificar actores y se desarrollan casos de uso de nivel alto para ayudar a definir el alcance del proyecto; 2. Fase de elaboración: se desarrollan los casos de uso más detallados, se utilizan para crear el plan para la fase siguiente. El producto de esta fase: escenarios primarios detallados y escenarios secundarios; 3. Fase de construcción: los casos de uso se utilizan como punto de partida para diseñar y para desarrollar planes de prueba; 4. Fase de transición: los casos de uso se pueden usar para desarrollar guías de usuarios y entrenamiento»*.

Un enfoque intermedio, más cercano al top-down, es el presentado por Constantine [Constantine 98] donde el proceso para la generación de casos de uso comprende: *«Identificar casos de uso candidatos basados en un modelo de roles de usuarios y brainstorming: generar una lista de casos de uso; Bosquejar un mapa de casos de uso basado en un entendimiento inicial (relaciones entre casos de uso); Desarrollar narrativas de casos de uso en una forma estructurada; Reducir casos de uso concretos a la forma esencial; Intercambiar entre narrativas y el mapa cuando sea necesario»*.

Por otro lado, el método de Jacobson [Jacobson 94] [Jacobson 94b] sigue un enfoque estrictamente bottom-up dado que: *«Primero describimos los casos de uso básicos y luego describimos cómo mejorarlos para permitir casos de uso más avanzados, algunos de los cuales pueden depender de los básicos. Los casos de uso abstractos deben evolucionar a partir de casos de uso concretos, no de la otra forma. Extensiones de asociación nos permiten capturar los requisitos funcionales de un sistema complejo, de la misma forma que aprendemos cualquier nuevo tema: Primero comprendemos las funciones básicas, luego introducimos complejidad»*.

Estudiando el Proceso Unificado [Jacobson 99] [Kruchten 04], se observa que aunque se consideran iteraciones y retrocesos – avances en el proceso, la estrategia subyacente para la construcción de casos de uso en el Flujo de Trabajo de Requisitos sigue una estrategia bottom-up, pues: *«Primero el analista de sistemas ejecuta la actividad de Encontrar Actores y Casos de Uso para preparar una primera versión del modelo de casos de uso, con los actores y casos de uso identificados»*. En esta actividad una vez identificados actores y casos de uso, éstos se describen brevemente y se construye el diagrama de casos de uso en su totalidad, que muestra las relaciones entre los casos de uso y los actores. Posteriormente sigue: *«El arquitecto identifica los casos de uso relevantes arquitectónicamente hablando, para proporcionar entradas a la priorización de los casos de uso»*. Luego: *«Hecho esto, los especificadores de casos de uso describen todos los casos de uso que se han*

priorizado. Más o menos en paralelo con ellos, los diseñadores de interfaz de usuario sugieren las interfaces de usuarios adecuadas para cada actor basándose en los casos de uso. Entonces el analista de sistemas re-estructura el modelo de casos de uso definiendo generalizaciones entre los casos de uso». Esta última actividad comprende detectar funcionalidades compartidas entre casos de uso, funcionalidades adicionales y opcionales, y otras relaciones entre casos de uso.

Gough et al. [Gough 95] siguen un enfoque cercano al propuesto en esta tesis, observando sus heurísticas: *«1. Creación de documentos en lenguaje natural: documentos de alcance del proyecto, documentos de necesidades del cliente, documentos de necesidades de servicio, análisis de competencia y análisis de mercado. Obtener especificaciones funcionales de un sistema existente, 2. Identificar los actores principales en el sistema, 3. Derivar los encabezados de escenarios, 4. Completar la descripción de escenarios»*. En las conclusiones establecen claramente que: *«La necesidad de proveer una vista a nivel alto de la interacción de escenarios como medio para destacar la complejidad de un gran número de escenarios»*.

Weidenhaupt et al. [Weidenhaupt 98] han encontrado en la práctica real cuatro tipos de procesos para la construcción de escenarios: i) Descomposición Top-down, ii) Desde escenarios de caja negra a escenarios de caja transparente[45], iii) Desde definiciones de escenarios informales a formales, y iv) Desarrollo incremental de escenarios. El primero, el segundo y el cuarto son enfoques top-down.

Se ha escrito bastante sobre top-down y bottom-up, pero se debe resaltar la visión de Michael Jackson sobre este tema. Jackson en [Jackson 95] analiza el enfoque top-down cuando se lo utiliza para obtener conocimiento de un problema a resolver, y describe dos dificultades básicas: *«La primera es que Top-Down obliga al ordenamiento más riesgoso posible en las decisiones. La decisión más grande es la subdivisión del problema completo: es la más grande en escala y la más grande en sus consecuencias. A pesar de ello, esta decisión se toma primero, cuando aún no se conoce nada y todo resta por descubrirse»*. Y *«La segunda dificultad es que el mundo real casi nunca tiene una estructura jerárquica. Por el contrario, hay muchas estructuras paralelas y superpuestas»*. Su consejo final acerca del uso del enfoque top-down para comprender un problema es simplemente: *«No lo use»*.

Con estos breves extractos de la literatura, se puede observar que no hay posturas concordantes sobre cómo construir escenarios y casos de uso. Se puede decir que, si la funcionalidad del sistema es bien conocida, una aproximación top-down podría utilizarse, mientras que, si no hay suficiente conocimiento, éste debería capturarse mediante una estrategia diferente.

[45] El término caja transparente se usa como sinónimo de caja blanca o caja de cristal.

2.8.3. Técnica de Inspección de Requisitos

Un aspecto de vital importancia, ya discutido en secciones previas, es cómo asegurar la calidad de los requisitos, plasmados éstos en distintos modelos y documentos, como por ejemplo los escenarios. Uno de los métodos más efectivos para lograr productividad con calidad al desarrollar software es utilizando un proceso de revisión. La idea es simple y varios reportes confirman su utilidad. La revisión es una vieja tradición en la industria de la publicación y ha estado en práctica por siglos. Algunas personas escriben y otras leen el documento escrito para corregir defectos y acomodar el texto según políticas predefinidas.

En la ingeniería de software, varios autores proponen revisiones: el ciclo autor / lector de SADT [Ross 77b], structured walkthrough [Yourdon 89b] [Gilb 89], revisiones de diseño [Parnas 85] e inspecciones [Fagan 76] [Fagan 86]. Sólo más recientemente [Ackerman 89] [Gilb 93] [Regnell 99c] [Laitenberger 00], los desarrolladores de software se han convencido que las revisiones son una necesidad. Aunque la mayoría de los procesos en uso actualmente basan el aseguramiento de calidad en el testeo.

Una lectura ad-hoc de escenarios encuentra defectos. Sin embargo, si hay pistas disponibles acerca de dónde se pueden localizar los defectos y cómo se pueden percibir, entonces la tarea de buscarlos se facilita y mejora la productividad de la lectura. Lectores experimentados encuentra más defectos que los novicios. En todas las áreas del conocimiento humano, la experiencia se condensa en herramientas, métodos, reglas o procedimientos para ayudar a personas con poca o sin experiencia a realizar tareas complejas.

Las revisiones se han formalizado aplicando diferentes procedimientos. Entre ellos, las inspecciones han surgido como una de las técnicas de aseguramiento de calidad más eficaces en la ingeniería de software [Laitenberger 00] [Gilb 93]. Fagan [Fagan 76] fue el inventor de la inspección de software como un proceso formal de detección de defectos en código fuente, compuesto de seis pasos bien definidos: Planeamiento, Apreciación global, Preparación, Reunión, Corrección y Seguimiento, y con roles específicos: inspector, productor, moderador y escriba.

Hoy en día las inspecciones se aplican a los documentos de requisitos con gran éxito [Martin 90b] [Schneider 92] [Porter 95] [Gough 95]. El uso real de la técnica de inspección en el campo ha conducido al desarrollo de varias variantes de inspecciones. Un ejemplo es la técnica N-fold [Martin 90b] [Schneider 92] que utiliza múltiples equipos de inspectores. Schneider informó que su investigación señaló que una repetición de nueve pliegues del proceso de inspección levantó la proporción de falta-detección de 35% (un único equipo independiente) a 78%. Kantorowitz et al. [Kantorowitz 97] propuso un modelo probabilístico simple para predecir la proporción de falta-detección basada en dos variables: nivel de especialización de los inspectores y el número de equipos de inspección.

Las técnicas de lectura para el paso de Preparación fueron clasificadas por [Regnell 99c] en cuatro grupos: Ad Hoc, Checklist, Basada en Escenarios[46] y Lectura Constructiva. Checklist es la propuesta más cercana a la de Fagan. La Figura 2-11 ilustra la taxonomía de inspecciones teniendo en cuenta el enfoque de lectura para el paso Preparación.

La *lectura Ad Hoc* se hace en una modalidad desestructurada, donde el inspector recibe muy poco apoyo para encontrar defectos. Esto no significa que los participantes de la inspección no escrutan el artefacto sistemáticamente. Sólo significa que ningún plan previo está disponible y todo se deja librado a la experiencia del inspector.

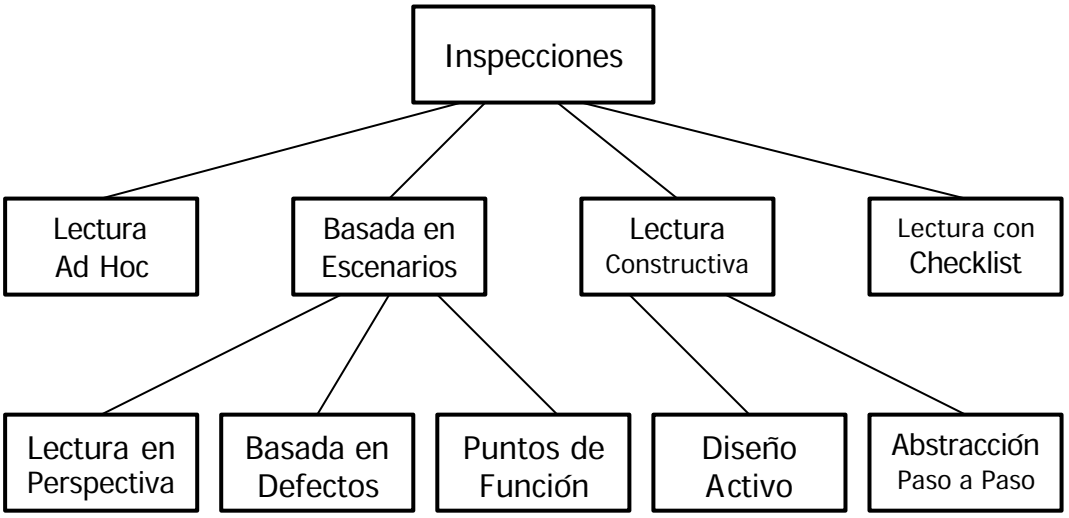


Figura 2-11. Taxonomía de inspecciones basadas en enfoques de lectura

La *lectura con Checklist* le da un apoyo fuerte al inspector en la forma de una lista de preguntas y controles que tienen que ser contestados y verificados en un cierto orden preestablecido. Este enfoque tiene algunas debilidades, como la falta de ayuda en comprender el artefacto bajo estudio y pobre adaptación a un ambiente de desarrollo dado.

La *lectura basada en Escenarios* mueve el centro de la inspección de la Reunión hacia el paso de la Preparación. Esta técnica da una guía detallada al inspector sobre cómo encontrar defectos específicos. Los ejemplos de esta variante son: Lectura en Perspectiva [Basili 96], Lectura basada en Defectos [Porter 95] y Lectura por Punto de Función [Cheng 96].

[46] Aquí el uso del término escenario (completamente diferente al de la sección anterior) se refiere a un meta-concepto que denota un procedimiento que un lector de un documento debe seguir durante la inspección. Cuando se habla entonces de "basado en escenarios", se está diciendo lo que Porter ha definido en [Porter 95]: "Escenarios - colecciones de procedimientos para detectar clases particulares de faltas".

La *lectura Constructiva* va más allá que meramente revisar un artefacto y producir una lista de defectos, pues durante la lectura el inspector produce una nueva representación del material bajo estudio, la que se analiza posteriormente. Las variantes de esta técnica son: Revisión con Diseño Activo [Parnas 87] y Abstracción Paso a Paso [Dyer 92].

Se llevaron a cabo varios experimentos para comparar diferentes técnicas de inspección aplicadas a documentos de requisitos: [Porter 95] [Basili 96] [Ciolkowski 97] [Fusaro 97] [Miller 98] [Porter 98] [Sandall 98] [Shull 98]. La Tabla 2-7 resume los resultados de estos experimentos, donde las inspecciones basadas en Escenarios realizaron una labor más eficiente que las Ad Hoc y Checklist. Así es que, las técnicas basadas en Escenarios frente a las Ad Hoc tuvieron mayor eficiencia en 10 experimentos y mejor que Checklist en 5 experimentos. Se debe tener en cuenta que no todos los experimentos efectuados compararon basado en escenarios contra Ad Hoc y/o Checklist.

Eficiencia de Lectura *basada en Escenarios*	**Lectura *Ad Hoc***	**Lectura *Checklist***
Mejor que	10	5
Igual que	4	5
Peor que	---	---

Tabla 2-7. Comparación de eficiencia de Lectura basada en Escenarios contra Ad Hoc y Checklist

En algunos experimentos, se siguió la experiencia del inspector más cuidadosamente [Porter 98] [Paech 05] y los resultados mostraron que la importancia de la definición del procedimiento disminuía a medida que aumentaba la experiencia del inspector, y que inspectores novicios podían adquirir rápidamente conocimiento sobre defectos típicos y aspectos de calidad. Un experimento [Regnell 99c] mostró en dos casos que no había diferencia entre la lectura basada en Defectos y la lectura en Perspectiva. Paech et al. [Paech 05] comprobaron que se detectaban en promedio más defectos usando la técnica de lectura basada en Escenarios frente a la lectura con Checklist, aunque acarreaba más tiempo de inspección.

En esta sección se da cuenta de la efectividad sostenida por las inspecciones. Últimamente se ha dedicado un notorio esfuerzo en producir variantes de esta técnica para verificar documentos de requisitos.

2.9. El framework: Client-Oriented Requirements Baseline

Esta tesis está enmarcada en la propuesta de Leite sobre el *Client-*

Oriented Requirements Baseline[47] (CORB) [Leite 95] [Leite 97], que apunta a modelar los requisitos externos de un sistema de software y su evolución. El CORB contiene descripciones sobre el dominio del problema y el artefacto de software a ser construido dentro de ese dominio. Es una estructura dinámica generada durante el proceso de IR, que evoluciona junto al proceso de desarrollo del software y que acompaña las tareas de mantenimiento. La idea de un CORB es fundamental para proveer las bases para gestionar los cambios en el proceso de evolución del software. El CORB es un ancla en el UdeD y como tal es un canal de comunicación con el mundo en donde el software operará.

La representación del CORB se basa en el LN, lo que facilita dicha comunicación, utilizando para ello dos modelos[48]:

i) *Léxico Extendido del Lenguaje*, un glosario que describe el lenguaje usado en el UdeD.
ii) *Escenarios*, descripciones de situaciones que muestran el comportamiento de la aplicación en un momento específico.

Adicionalmente, el CORB se apoya en dos subsistemas ortogonales a los dos modelos anteriores:

i) *Subsistema de Hipertexto*, que facilita el acceso a la información que provee el CORB, integrando el modelo léxico y el modelo de escenarios mediante vínculos internos dentro de cada modelo junto con vínculos entre ambos modelos. Las descripciones de los escenarios tienen un fuerte vínculo con el LEL, ya que adoptan este léxico como referencia.
ii) *Subsistema de Versionado*, que permite llevar la historia de los cambios en el LEL y en el conjunto de Escenarios durante el desarrollo y el mantenimiento del software. Este subsistema muestra la evolución del sistema de software, garantizando la trazabilidad de los requisitos.

Se debe enfatizar la importancia de la capacidad de referencias cruzadas entre el CORB y los modelos y/o artefactos construidos en otras etapas del proceso de software, así como la existencia de referencias cruzadas dentro del propio CORB. De esta forma, el CORB provee los medios para la trazabilidad de los requisitos.

Si se toma como ejemplo los escenarios dentro del CORB, éstos son descripciones evolutivas de las situaciones del UdeD. Por ende, comienzan describiendo situaciones en el UdeD, y luego evolucionan a lo largo del

[47] *Andarivel Base de Requisitos* sería la traducción de *Requirements Baseline* que se prefiere omitir al no representar con precisión el concepto, cuyo origen proviene de la Gestión de Configuración [Babich 86]. En el área de Gestión de Configuración del Software al Requirments Baseline también se lo conoce como Baseline Funcional [Scott 00].

[48] La versión original del CORB [Leite 95] incluye un tercer modelo Básico, basado en el modelo de entidad-relación, que expresa los requisitos externos propuestos por los clientes y usuarios, pero que fuera abandonado en posteriores refinamientos de la estrategia.

proceso de construcción del software, describiendo cómo estas situaciones cambian en el UdeD, tanto por la inclusión del software mismo como por la naturaleza cambiante del propio UdeD.

El CORB promueve un proceso de software dirigido por los requisitos, es decir, independiente del método adoptado de construcción y mantenimiento del software.

Esta tesis presenta en los siguientes capítulos reformulaciones, ampliaciones, mejoras y precisiones a la propuesta de Leite en [Leite 97].

Capítulo 3
SDRES, una Estrategia en la Ingeniería de Requisitos

«There is no sense in being precise about something when you don't know what you are talking about. »

John Von Neumann, citado en M. Jackson, "Software Requirements & Specifications", 1995, página 65

3.1. Introducción

De lo expuesto en los dos capítulos precedentes, la estrategia en la IR que se expone en esta tesis se sustenta en dos pilares base: la evolución y la comunicación. Para lo primero, la estrategia se apuntala en la técnica de escenarios, la cual permite representar fácilmente la evolución que sufre el UdeD a lo largo del tiempo [Leite 04b]. Para lo segundo, esta estrategia se apoya en la construcción de un glosario del UdeD, enfatizando el uso de dicho vocabulario tanto en los modelos y documentos que se generan como en cualquier presentación escrita u oral realizada por los ingenieros de software. Adicionalmente, los escenarios también son un medio de comunicación atractivo pues son sencillos de comprender por parte de los clientes y usuarios, y favorecen la colaboración entre todos los involucrados para la definición de los requisitos.

La comunicación es una actividad compleja pero fundamental en cualquier relación humana: permite que dos o más personas con sus propios intereses puedan entenderse. Durante todo el proceso de desarrollo de software, la comunicación es un elemento indispensable donde cada involucrado debe hacer uso de ella a diario [Gause 89]. Cuando los ingenieros de software se comunican con los clientes y usuarios, los primeros usan un vocabulario orientado a la computación, mientras que los clientes y usuarios usan la terminología proporcionada por su ambiente [Loucopoulos 95]. Una comunicación basada en léxicos diferentes tiene una alta probabilidad de fracaso. Aunque usando el mismo idioma cada parte tiene su propio léxico; esto significa que el problema de comunicación es intrínseco al contexto social y cultural del idioma [Malinowski Rubio 01]. Un ejemplo típico es cuando se emplea el mismo término pero con significados diferentes. Si los ingenieros de software no perciben la ambigüedad y no están conscientes que algunos términos de los clientes y usuarios pueden tener un significado diferente en su contexto particular de trabajo, entonces los ingenieros dejarán que se introduzcan errores ocultos en su comprensión del UdeD.

Como la mayoría del conocimiento sobre un dominio del problema se expresa en LN [Loucopoulos 95], el uso de un enfoque basado en representaciones en LN para elicitar y comprender los requisitos incrementa la probabilidad de éxito. Así, como muchos otros autores, en esta tesis también se adhiere a una estrategia basada en LN en la IR. Una revisión hecha por

Rolland et al. [Rolland 98] muestra que de doce enfoques propuestos en la literatura basados en escenarios, todos ellos usan una notación de texto para describir escenarios, que en algunos casos se combinan con otros medios como gráficos o imágenes. Según el estudio de Weidenhaupt et al. [Weidenhaupt 98] la mayoría de las representaciones de casos de uso y escenarios usadas en la práctica real se expresan en LN: trece de un total de quince proyectos estudiados. En un estudio más reciente sobre la práctica en IR [Neill 03], el 51% de las organizaciones (sobre un total de 194) usa representaciones informales (por ejemplo, LN) y el 27% semi-formales. Los modelos formales se usan apenas el 7%.

Por otro lado, más veces de lo esperado, la validación de requisitos se ha vuelto una tarea compleja, principalmente debido al tipo de representaciones formales o gráficas de los modelos, que requieren en muchos casos clientes y usuarios con habilidades especiales para entenderlos. Como remarcó Sommerville en [Sommerville 01], la validación de requisitos raramente descubre todos los problemas de requisitos, arrastrando éstos a las subsiguientes fases del desarrollo. El uso de representaciones basadas en LN ayuda a la validación de requisitos, la cual se hace más eficiente cuando la representación se expresa usando el vocabulario de los propios clientes y usuarios.

Los enfoques basados en LN ciertamente tratan con dos desventajas conocidas: i) ambigüedad e imprecisión [Jackson 95] [Sommerville 01], y ii) dificultad en lograr una verificación automatizada. Esta tesis intenta demostrar que ambos obstáculos pueden reducirse fuertemente, reforzando las bondades de este tipo de enfoques, que como ya se mencionó son: i) la facilidad de comprensión y validación de los modelos y ii) la necesidad de usuarios e ingenieros de requisitos con menos experiencia formal. En función de la experiencia obtenida, en esta tesis se considera que un enfoque basado en LN puede además reducir los defectos en los requisitos usando métodos y herramientas apropiadas. Las especificaciones formales son precisas y no ambiguas facilitando la detección temprana de ciertos defectos[49], pero son casi incomprensibles para los clientes y usuarios, quienes son renuentes a usarlas como contratos [Sommerville 01].

Dan Berry en [Berry 04] clasifica las ambigüedades en: 1) ambigüedad léxica (una palabra con varios significados), 2) ambigüedad sintáctica o ambigüedad estructural (una oración con más de una posible estructura), 3) ambigüedad semántica (una oración con más de una manera de leerse), 4) ambigüedad pragmática (una oración cuyo significando depende del contexto), 5) vaguedad (una oración con falta de detalles) y 6) error idiomático (una oración con errores como la puntuación). El uso de un glosario reduce la ambigüedad léxica y pragmática y, en menor medida, la vaguedad. El propio proceso de creación del glosario (ver capítulo 4) actúa activamente contra estos tres tipos de ambigüedad. La ambigüedad sintáctica, ambigüedad

[49] A pesar de las virtudes de las especificaciones formales, éstas no ayudan en la elicitación de información ni en la detección de ciertos defectos como omisión de información (información no elicitada, descubrimiento tardío), mala interpretación del UdeD, información errónea proveniente del UdeD, entre otros.

semántica y error del idioma pueden aparecer luego en otros documentos independientemente del uso de un glosario. El control de tales ambigüedades está fuera del alcance y posibilidades de un glosario dado que son causadas por los propios mecanismos de creación de las oraciones.

Respecto a la dificultad de verificación de modelos en LN, la estrategia propone el uso de la técnica de inspecciones [Fagan 74], que es de alta efectividad en la detección de defectos, para comprobar los modelos producidos.

La estrategia que se presenta en esta tesis, denominada SDRES, se sustenta en la construcción y uso de dos modelos basados en LN: el Léxico Extendido del Lenguaje, un glosario especial del vocabulario del dominio de la aplicación, y Escenarios, descripciones de situaciones en el UdeD. El principal propósito de este léxico es la captura del vocabulario del dominio de la aplicación, mientras que la comprensión de la funcionalidad y características del UdeD se obtienen por medio de los escenarios. Cada escenario se describe usando los términos definidos en el LEL. Los escenarios sirven tanto para describir situaciones que ocurren actualmente en el UdeD: "situaciones observadas o reales", como así también, situaciones que ocurrirán cuando se incorpore el nuevo artefacto de software en el UdeD: "situaciones proyectadas o esperadas"[50]. Esta incorporación traerá inevitablemente un cambio en el UdeD. En esta tesis, se pretende destacar que los escenarios son descripciones de situaciones que evolucionan en el UdeD. Los escenarios comienzan describiendo situaciones en el UdeD y luego evolucionan a lo largo del proceso de desarrollo del software, describiendo cómo estas situaciones cambian en el UdeD [Leite 04b].

Ambos conjuntos de escenarios persisten a lo largo del desarrollo de software, ambos describen situaciones en el UdeD utilizando el vocabulario del UdeD, la única diferencia aunque sustancial entre ambos conjuntos es el contexto que describen. El CORB puede contener ambos conjuntos de escenarios junto con otros artefactos de requisitos que se deseen crear, tal es el caso del modelo léxico y del SRS.

En resumen, SDRES es un enfoque dentro de la IR dirigido por modelos basados en LN, que produce un glosario del UdeD y aplica la técnica de escenarios con el propósito de obtener conocimiento acerca del problema y capturar los requisitos del software. Los modelos que se construyen no sólo sirven para registrar información y facilitar el análisis de la misma, sino que poseen la gran ventaja de motivar la elicitación de información. Es decir, el LEL y los escenarios no son meros instrumentos contenedores organizados de información sino que son en sí mismos promotores de la captación de información, estimulando además la introspección de los ingenieros y facilitando la transmisión de información por parte de los clientes y usuarios. Ambas representaciones no intentan reemplazar el documento de especificación de requisitos, el objetivo de ellas es ayudar a la producción de especificaciones. Además constituyen el *requirements baseline* para la gestión

[50] Esta distinción entre situaciones presentes y situaciones proyectadas no está presente en la propuesta original de Leite [Leite 95] [Leite 97], y tampoco en sus versiones más recientes [Breitman 05].

de cambios, según lo expresado en la sección 2.9.

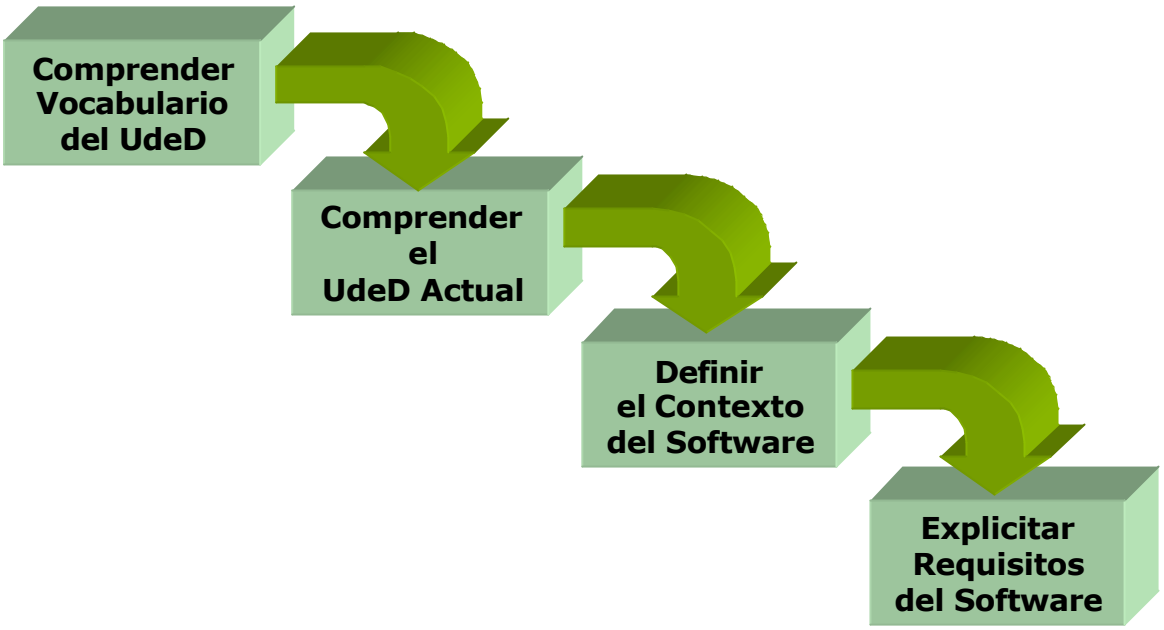


Figura 3-1. SDRES, una estrategia en la IR

Como se esquematiza en la Figura 3-1, SDRES presenta un refinamiento del proceso de definición de requisitos, que básicamente consiste en cuatro etapas principales: i) comprender el vocabulario del UdeD, para ello se construye un LEL del UdeD; ii) comprender el UdeD actual, para ello se construye un conjunto de escenarios representando situaciones actuales (que se denominarán Escenarios Actuales); iii) definir el contexto del sistema de software, para ello se evoluciona el conjunto de escenarios previos en otro conjunto que represente las situaciones futuras que ocurrirán cuando se implante el nuevo sistema de software (conjunto que se denominará Escenarios Futuros); y iv) explicitar los requisitos del software[51], derivándolos de los Escenarios Futuros y generando un documento con ellos.

Las dos primeras etapas referidas al UdeD actual (ver capítulos 4 y 6) se basan en un enfoque presentado en [Leite 97], donde en esta tesis se ofrecen algunas mejoras y formalizaciones, extendiendo su uso en la práctica [Hadad 99] [Hadad 99b] [Leite 00] [Leite 05]. Las dos últimas etapas son las que se presentan en esta tesis cuyos avances se observan en [Doorn 02] [Hadad 03].

Como consecuencia de lo antes mencionado, SDRES apunta a obtener la mayor comprensión del dominio de la aplicación con el fin de proponer la mejor solución negociada, estableciendo un puente entre las necesidades de los clientes y usuarios, y la solución de los diseñadores (ver sub-secciones 2.2.1 y 2.2.2).

[51] Explicitar requisitos de software en un documento es una actividad que no está presente en ninguna de las propuestas de Leite (ver Bibliografía de esta tesis) y que además no se ha encontrado en propuestas de otros autores.

La Figura 3-1 sólo intenta presentar las etapas de la estrategia y por ello no presenta los reciclos que existen entre las etapas por una mejor comprensión del UdeD. Este detalle puede observarse en la Figura 3-2.

Cabe destacar, que por razones meramente explicativas, se muestra a la estrategia siguiendo un modelo de cascada, pero se verá en la sección 3.4 cómo puede aplicarse y adaptarse a múltiples modelos de proceso.

En las siguientes secciones se describen y justifican cada una de las etapas que componen esta estrategia en la IR, las que luego se detallarán en los capítulos 4, 5, 6, 7 y 8, y cuyos resultados aplicados a diversos casos se exponen en el apéndice D. Se presenta además en este capítulo la evolución que sufren estos modelos, una guía para la aplicación de la estrategia y un sistema de versionado para estos modelos.

3.2. Etapas de la Estrategia

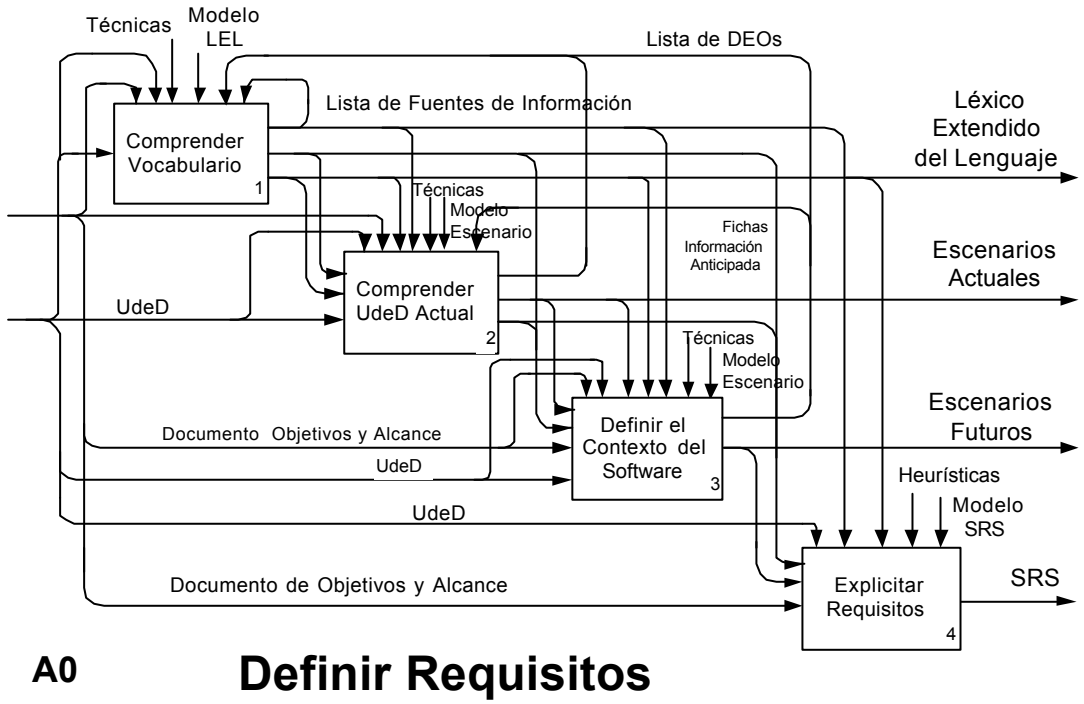


Figura 3-2. SADT de la estrategia en la IR

En la Figura 3-2 se muestra mediante el modelo SADT las cuatro etapas en las cuales se desarrolla la estrategia y los productos obtenidos en cada una. En esta figura se han incorporado los reciclos que ocurren por mejoras en la comprensión del UdeD, donde se vuelve a refinar el LEL, los Escenarios Actuales y / o los Escenarios Futuros. En la cuarta etapa no hay un feedback a etapas anteriores, porque es una etapa que trata sobre reorganizar la información ya elicitada y modelada en la etapa anterior.

Además, se visualiza en la Figura 3-2 un producto intermedio de las tres primeras etapas, denominado Fichas de Información Anticipada (ver sección 4.5), donde se vuelca todo lo que se elicita y es importante pero no corresponde al modelo que se está construyendo en dicha etapa sino que es información útil para alguna etapa posterior.

A continuación se describe para cada etapa: el objetivo que cumple, los modelos que se generan, una breve descripción del proceso que se sigue y su importancia en la estrategia.

1ª Etapa	**Comprensión del Vocabulario del UdeD**
	Se trata en detalle en el capítulo 4.
Objetivo	
	Conocer el vocabulario empleado en el UdeD.
Productos	
	Léxico Extendido del Lenguaje Fichas de Información Anticipada
Breve Descripción	
	El proceso que se sigue en esta etapa comienza elicitando información general del UdeD para generar una lista de términos relevantes o de uso específico en dicho UdeD. Estos términos luego se irán definiendo con la captura de más información. Este proceso no mira específicamente a todo el UdeD ni ahonda en cada detalle de la funcionalidad del problema, por el contrario avanza y retrocede de las ideas generales al conocimiento detallado solamente para comprender el vocabulario usado en el UdeD y para registrarlo en un glosario denominado LEL. En esta actividad debe extremarse el cuidado en no insertar términos o significados que el ingeniero de requisitos considera que debieran usarse, pero que efectivamente no son usados por los clientes y usuarios. Es común obtener durante esta etapa más información de la realmente necesaria para construir el LEL. Para evitar la pérdida de esta información, se la vuelca en una Ficha de Información Anticipada.
Justificación de la etapa	
	El LEL es un modelo de gran utilidad a lo largo de todo el proceso de desarrollo del software, pues facilita la comunicación escrita y oral entre los involucrados y reduce la ambigüedad en

	los documentos producidos durante la IR (Escenarios Actuales, Escenarios Futuros y SRS), como así también en otros artefactos generados posteriormente que incluyan alguna descripción textual y que deban ser comprendidos por los clientes y usuarios.

2ª Etapa	**Comprensión del UdeD actual**
	Se trata en detalle en el capítulo 6.
Objetivo	
	Conocer el UdeD tal cual es en la actualidad.
Productos	
	Escenarios Actuales Fichas de Información Anticipada
Breve Descripción	
	El proceso en esta etapa comienza con la construcción de escenarios siguiendo las heurísticas que extraen información del LEL y se obtienen escenarios cuyo nivel de detalle es mayor que el encontrado en el LEL. La información adicional se obtiene principalmente del UdeD. Durante este paso algunas partes comunes de diferentes escenarios se factorizan para crear subescenarios. Los subescenarios también son generados cuando uno o más episodios de un escenario merecen un tratamiento independiente. Otros escenarios se descomponen en uno o más escenarios para facilitar su comprensión. Estos últimos pasos muestran un comportamiento descendente, así esta parte del proceso puede verse como top-down. En este punto, surge como una fuerte necesidad de los ingenieros el tener una visión global de los escenarios y un nuevo paso, ahora bottom-up, es requerido. En este paso se construyen los escenarios integradores, mencionando en sus episodios situaciones concretas descriptas por los escenarios obtenidos previamente. Por ende, el enfoque para la comprensión del UdeD actual no es ni top-down ni bottom-up, ya que no se mira todo el UdeD en forma global construyendo unos pocos escenarios generales para luego refinarlos progresivamente, pero tampoco se procede a la búsqueda de episodios que adecuadamente sistematizados den lugar a escenarios, los que a su vez se puedan integrar en escenarios de mayor alcance. El enfoque utilizado es considerado middle-out [Hadad 99] porque partiendo

de un nivel medio avanza y retrocede de lo particular a lo general aprovechando las ventajas de los enfoques top-down y bottom-up. Se aplica el enfoque top-down cuando las situaciones identificadas utilizando el LEL son refinadas y detalladas mediante escenarios. Es bottom-up cuando diferentes situaciones se reúnen en procesos o secuencias más amplias denominados escenarios integradores. Es decir, la propuesta middle-out está basada en la simple idea de aprovechar al máximo el conocimiento del problema bajo estudio, antes de tomar una buena decisión sobre la solución o alternativas de solución a proponer.

Los escenarios son descriptos utilizando el vocabulario del UdeD, esto implica entonces no sólo obtener una mayor comprensión del UdeD sino también de su vocabulario, lo que provoca inevitablemente la evolución del LEL. Esto no es porque cambia la forma de expresarse en el UdeD sino que existe una mejor captación por parte de los ingenieros de requisitos.

En esta etapa se debe evaluar si alguna Ficha de Información Anticipada contiene información sobre las actividades actuales que no se incorporó en el LEL y que corresponde incluir en los escenarios actuales.

Al igual que en la etapa previa, pero en mayor medida, la captura de información que excede la realidad actual es un hecho frecuente, y para evitar la pérdida de dicha información anticipada o distraer al ingeniero en el análisis de la misma, se vuelca dicha información en Fichas de Información Anticipada.

Justificación de la etapa

Algunas propuestas simplifican o descartan la necesidad de modelar el contexto actual y/o de conocer el vocabulario utilizado en él. Sin embargo, la construcción de los escenarios actuales es una tarea básica para permitir una adecuada pre-trazabilidad de los requisitos, la cual hoy en día es considerada una medida de calidad de los sistemas y exigida por muchos estándares de desarrollo de software.

Por otro lado, muchas veces los desarrolladores se desempeñan bajo la creencia de que conocen o comprenden ese contexto y/o ese vocabulario, pero en la mayoría de los casos tienen un conocimiento parcial y, lo que es peor aún, distorsionado de la realidad y de su lenguaje. Esto es especialmente cierto en proyectos de gran envergadura y/o con un grado importante de complejidad.

Si el ingeniero de requisitos no conoce nada o muy poco del UdeD entonces procurará informarse en forma apropiada,

pero si tiene algún conocimiento entonces tiene una tendencia natural de asociar la información que recibe con sus conocimientos previos, cometiendo en muchos casos errores de interpretación o "llenando huecos" con hipótesis y detalles de su propio conocimiento previo que pueden conducirlo a errores. Desafortunadamente estas circunstancias son las más frecuentes.

Adicionalmente, esta etapa permite identificar problemas y oportunidades de mejoras en los procesos del negocio actual. Por lo tanto, es una etapa esencial cuando se supone que los procesos del negocio son ineficientes, por ejemplo, cuando se está admitiendo una reingeniería de los procesos.

3ª Etapa	**Definición del Contexto del Software**
	Se trata en detalle en el capítulo 7.
Objetivo	
	Definir los cambios necesarios en los procesos del negocio para hospedar y beneficiarse con el software a construir.
Productos	
	Escenarios Futuros Fichas de Información Anticipada
Breve Descripción	
	Se comienza seleccionando la estrategia de construcción de escenarios propiamente dicha, que depende de los objetivos del sistema y del grado de mejoras en los procesos del negocio esperado. En el caso de un bajo nivel de cambio en los procesos del negocio, se sigue una estrategia donde se establecen mejoras a cada escenario actual convirtiéndolo en un escenario futuro, es decir se estudia la jerarquía de escenarios actuales siguiendo una modalidad "post-order" (en profundidad). En el caso de grandes cambios en los procesos del negocio, por el contrario, los escenarios actuales de mayor nivel en la jerarquía sufrirán una alta re-estructuración y así hacia abajo en dicha jerarquía en función de los objetivos del sistema, es decir se sigue una estrategia "pre-order" (en amplitud). Para niveles medios de cambio, la estrategia a seguir es una combinación de las anteriores. Durante esta etapa se tiene en consideración la información registrada en las Fichas de Información Anticipada,

además de ser necesaria la elicitación de más información en el UdeD. A través de la construcción de los escenarios futuros se presentan propuestas alternativas de solución que son negociadas con los clientes y usuarios.

Cabe mencionar que es una etapa donde se intensifica la elicitación de RNF, algunos de los cuales por sus características generales (de aplicación a todo el sistema o a una buena parte del mismo) no pueden vincularse a un EF específico, dado lo cual se los registra en las Fichas de Información Anticipada para su inclusión posterior en el SRS.

Justificación de la etapa

Los escenarios futuros albergan los requisitos del software en una forma altamente legible y comprensible por los clientes y usuarios, lo que facilita su negociación y priorización.

Dada su rica expresividad, son un medio que facilitan la elicitación de requisitos, la negociación de los mismos y su validación. Estimulan la imaginación, facilitando la generación de propuestas por parte de todos los involucrados. Permiten presentar sin grandes costos (por su facilidad de construcción) distintas alternativas de solución a los problemas y necesidades manifestadas en el UdeD.

A través de los escenarios futuros los clientes y usuarios pueden priorizar los requisitos, evaluando la criticidad de las situaciones futuras y otorgándoles a éstas prioridades.

Los escenarios futuros sirven de ancla para la pre y post trazabilidad, permitiendo el rastreo de los requisitos hacia sus orígenes (vinculando los escenarios futuros con los escenarios actuales y con el LEL) y hacia el diseño y el código. Son un medio que facilitan la gestión de los cambios en los requisitos a lo largo del ciclo de vida del software.

4ª Etapa	**Explicitar los Requisitos del Software**
	Se trata en detalle en el capítulo 8.
Objetivo	
	Establecer los requisitos del software en el contexto bajo estudio.
Productos	

	Documento de Definición de Requisitos
Breve Descripción	
	Una vez acordado el conjunto de escenarios futuros que incluirán las funcionalidades y características del sistema de software, se extraen de dicho conjunto los requisitos funcionales y no funcionales del software. Se redacta el Documento de Definición de Requisitos, que puede ir desde una simple lista de requisitos hasta un documento más formal basado en algún estándar que la organización utilice o se proponga de acuerdo al proyecto en particular.
Justificación de la etapa	
	La producción del documento SRS es fácilmente derivable de los escenarios futuros y además, dado que éstos son previamente verificados y validados y anclados en el LEL, se asegura la obtención de un SRS consistente, no ambiguo, completo, correcto, entendible, rastreable, verificable y abstracto[52] (ver sub-sección 2.6.3: Propiedades de los requisitos). La trazabilidad de los requisitos puede también manejarse a través de este documento, que facilita la individualización de los requisitos.

3.3. La evolución de los modelos

3.3.1. Tipos de evolución en SDRES

Como ya se ha mencionado en el capítulo 1, los cambios en los requisitos durante el ciclo de vida del software son una realidad y, por lo tanto, SDRES debe afrontarlos.

Respecto a la propuesta original de Leite [Leite 95] [Leite 97], en esta tesis se consideran dos evoluciones concurrentes de los escenarios.

Una evolución tiene causas que corresponden a la aparición de cambios en los requisitos, ya sean éstos cambios genuinos en el UdeD, cambios en las expectativas e intereses de los clientes y usuarios, o mejoras en la comprensión de los requisitos (correcciones debido a malas interpretaciones, omisiones u otros defectos atribuibles a las fuentes de información mismas o a

[52] Será abstracto en la medida que no se incorporen requisitos dependientes de implementación por pedido expreso de los clientes y usuarios o por acatar políticas y normas vigentes en la organización o gubernamentales.

los propios ingenieros de requisitos).

Existe otra evolución en los escenarios cuya causa es interna a la estrategia propuesta en la IR. Ella consiste en la transformación de los escenarios actuales en escenarios futuros, imaginando lo esperado para el futuro según los objetivos fijados y el estado actual.

En esta tesis, se considera la primera evolución en forma concurrente para dos conjuntos de escenarios. Los escenarios que reflejan situaciones actuales van cambiando a lo largo del desarrollo de software pues i) el UdeD cambia y ii) se alcanza un mejor conocimiento del mismo. En simultáneo también los escenarios de situaciones proyectadas sufren modificaciones como consecuencia de i) cambios en el UdeD que obliga también a cambiar lo esperado para el futuro, ii) cambios en las expectativas e intereses de los clientes y usuarios, y iii) mejora en el conocimiento adquirido.

La gestión de requisitos trata esta evolución, la cual debe ser tenida en cuenta no sólo para los escenarios sino para todos los artefactos producidos durante el desarrollo del software. Con frecuencia una modificación en algún artefacto provocará una cascada de cambios en los artefactos derivados de éste. Por ejemplo, un cambio genuino del UdeD puede desencadenar en modificaciones al LEL, a los escenarios de situaciones actuales, a los escenarios de situaciones proyectadas, y a otros modelos de diseño y de implementación según el estado de avance del proyecto. Pero también el estudio de un modelo de diseño o de implementación puede dar origen a una duda o a una nueva percepción del UdeD que afectará a los escenarios de situaciones actuales y a los escenarios de situaciones proyectadas.

Ambas evoluciones comparten la característica de ser cuasi-continuas, en el sentido que habitualmente involucran muchas modificaciones de una granularidad media o menor.

Hay un segundo tipo de evolución, que es básicamente discreta y produce un gran cambio en un período breve de tiempo. Este tipo de evolución tiene lugar cuando todo el sistema o parte del mismo se pone en servicio, en este caso básicamente los escenarios futuros se convierten en escenarios actuales, pues el UdeD fue alterado a consecuencia de la incorporación del software. Posteriormente pueden o no surgir nuevos escenarios futuros por mantenimiento o ampliación del sistema. La puesta en operación de un sistema de software provoca grandes cambios en el UdeD, pudiendo incluso llegar a alterar el vocabulario empleado en él.

Ambos tipos de evoluciones están lejos de ser independientes, sino que por el contrario, se entremezclan de forma compleja y muchas veces impredecible. Es por ello que la estrategia debe tratarlas adecuadamente.

En la Figura 3-3 se presenta cómo evolucionan los modelos del proceso de desarrollo debido a una modificación genuina del UdeD que provoca una variación en el proceso del negocio actual. En la Figura 3-4 se presenta una evolución debido a un cambio en las expectativas de los clientes y usuarios. En

la Figura 3-5 se representa el impacto que produce una mayor precisión en la percepción que el grupo de desarrollo adquiere respecto a las expectativas de los clientes y usuarios, en este ejemplo dicha mejora se alcanza durante el diseño. En estas tres figuras, las líneas sólidas representan las evoluciones directas a partir de la causa primaria, mientras que las líneas punteadas representan las evoluciones que se desencadenan como consecuencia de aquellas. En ninguna de estas tres figuras se ha mostrado una evolución del LEL, pero como ya se ha mencionado, ésta es perfectamente admisible aunque menos frecuente. En la práctica la evolución de los modelos se puede dar concurrentemente por todas estas causas, obligando a que la gestión de requisitos sea una actividad imperativa en todo proceso de desarrollo de software.

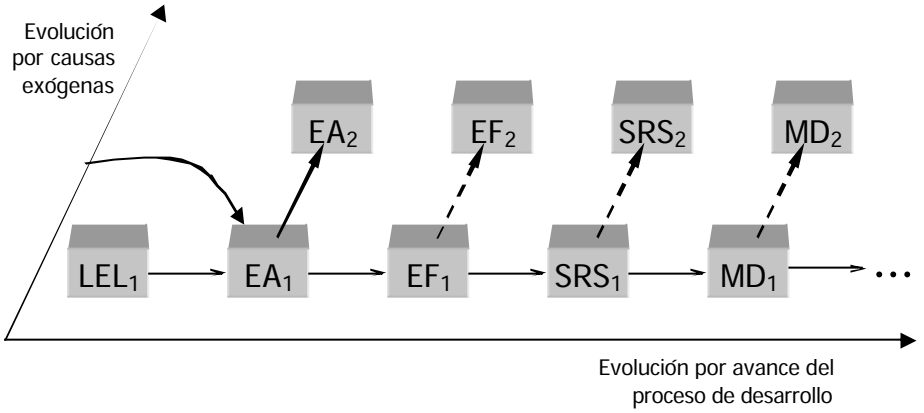


Figura 3-3. Evolución por cambios genuinos en el UdeD

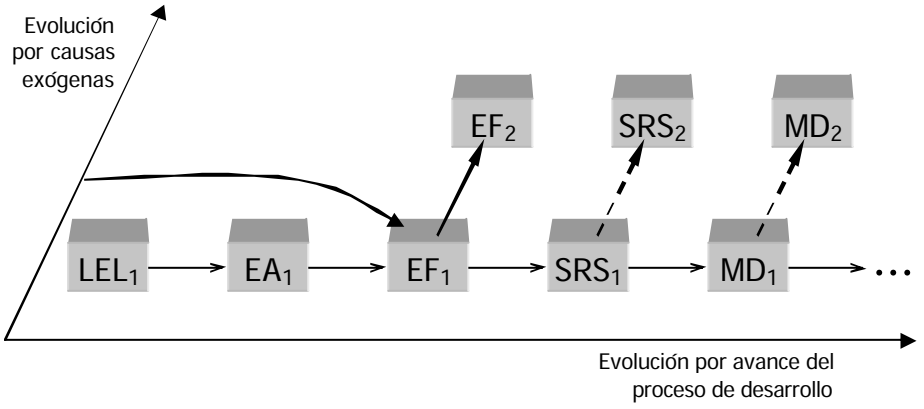


Figura 3-4. Evolución por cambios en las expectativas de los usuarios

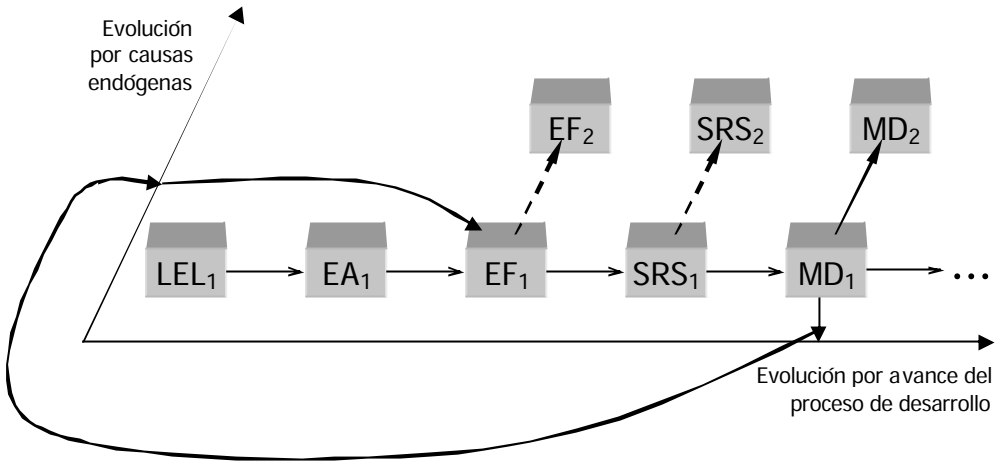


Figura 3-5. Evolución por mejora en la comprensión de las expectativas

Las Figuras 3-6 y 3-7 muestran la evolución que sufren los escenarios a consecuencia del cambio en el UdeD por la instalación parcial o total del sistema de software. En el caso de la puesta en operación parcial, esto implica que parte del conjunto de escenarios futuros fueron implementados y representan la nueva realidad actual del negocio, por ende, pasan a ser considerados como parte del conjunto de escenarios actuales. Del conjunto de escenarios actuales previo, existe una parte que ya no representa más lo actual y, por lo tanto, se descartan del conjunto de escenarios actuales y se suelen conservar con fines históricos. Luego, el conjunto de escenarios futuros fue reducido, restando menos tareas de desarrollo, y convirtiéndose en un nuevo conjunto de escenarios futuros. En el caso de la puesta en operación total del software, esto indica la finalización del proceso de desarrollo, donde ahora todo el conjunto de escenarios futuros representan la situación actual del UdeD, habiendo entonces un traslado del conjunto de escenarios futuros como un nuevo conjunto de escenarios actuales, y el conjunto previo de escenarios actuales pasará a formar parte de la historia del proyecto junto con el conjunto previo de escenarios futuros y restantes modelos generados. Posteriormente podrá surgir la necesidad de mantenimiento del sistema de software que involucrará reiniciar el ciclo de desarrollo, pero contándose ya con el LEL y el conjunto de escenarios actuales. La evolución presentada en estos dos casos de puesta en servicio puede percibirse como ingenua frente a la realidad que involucra una instalación parcial o total de un sistema de software, pues con frecuencia la propia instalación provoca la necesidad de algunos cambios menores o ajustes en el software debiéndose entonces trasladar estos retoques al nuevo conjunto de escenarios actuales de corresponder. Análogamente, puede ocurrir que la incorporación del software provoque un cambio en el vocabulario del UdeD (ver sub-sección siguiente), obligando a la actualización también del LEL. Esta evolución del vocabulario usualmente no es percibida inmediatamente a la instalación sino una vez asentado el software como un miembro más del UdeD.

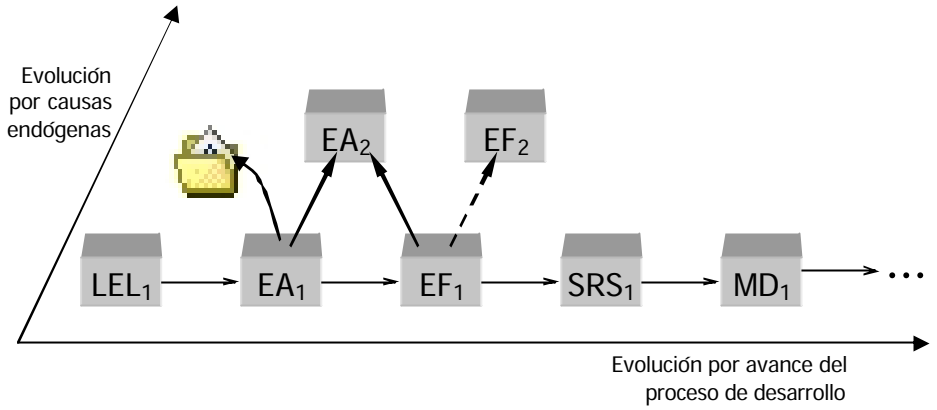


Figura 3-6. Evolución por puesta en servicio parcial del software

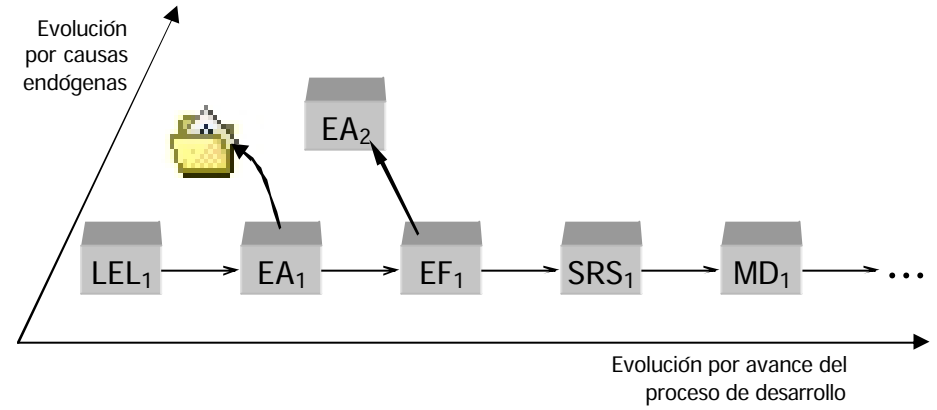


Figura 3-7. Evolución por puesta en servicio total del software

3.3.2. Gestión de la evolución de los modelos

El CORB puede estar constituido por el Léxico Extendido del Lenguaje (LEL), los Escenarios Actuales (EA), los Escenarios Futuros (EF) y el Documento de Definición de Requisitos (SRS). Cabe cuestionarse la necesidad y la oportunidad de mantener actualizados todos estos modelos y documentos a lo largo del ciclo de vida del software, dado el insumo de tiempo y recursos que implica.

En el otro extremo, se puede considerar como necesario y suficiente sólo mantener el conjunto de EF, y que a través de éstos pueda llevarse a cabo una adecuada gestión de cambios en los requisitos, dado que podría pensarse que una vez que se ha comprendido el UdeD ya no es necesario mantener ni el glosario ni los EA, y que el SRS puede derivarse de los EF con un mínimo de esfuerzo, siempre y cuando éstos contengan todos los requisitos del software.

Cabe aclarar que las Fichas de Información Anticipada son un producto

intermedio de registro auxiliar de información y por ende no se incluyen en el CORB.

Para mantener cada modelo y documento generado durante el proceso de la IR, se debe tener en cuenta los distintos tipos de mantenimiento según el origen de dichos cambios. Leite en [Leite 91] presenta cuatro tipos de mantenimiento de software, que pueden perfectamente considerarse para el caso de artefactos de IR, y son:

- ⇨ **Correctivo**: Corrección de errores originados en el proceso de producción.
- ⇨ **Evolutivo**: Agregado de nuevas funcionalidades exigidas por el ambiente externo.
- ⇨ **Adaptativo**: Incorporación de cambios originados en cambios de la plataforma de soporte de hardware o de software.
- ⇨ **De soporte**: Incorporación de modificaciones originadas en necesidades de eficiencia del software.

Esta tesis propone diferentes alternativas para el CORB (ver sección siguiente) en función de la complejidad del contexto, de la envergadura del proyecto, y de las características de la organización misma. Estas alternativas se discutirán en el contexto de los tipos de mantenimiento mencionados.

¿Por qué mantener el LEL?

A pesar que el vocabulario empleado en el dominio de una aplicación suele ser más estable que los procesos del negocio, el LEL evoluciona a medida que se tiene un mejor entendimiento del vocabulario y también a consecuencia de cambios en el UdeD que alteran aunque en menor medida el vocabulario con el que se expresan los clientes y usuarios (ver sub-sección 4.3.3: Evolución del LEL y sub-sección 4.4.2: Uso del LEL en otras estrategias). Se debe considerar además que la inserción de un software en dicho dominio también puede alterar el vocabulario del UdeD.

Si la documentación, modelos e incluso el código mismo generados durante el desarrollo del software utilizan en sus descripciones el vocabulario del UdeD, cambios en el lenguaje deberían ser trasladados al LEL para mantener una vinculación perenne entre este modelo y todo aquello que dé soporte al software. Dada la estabilidad natural del lenguaje en un contexto, el mantenimiento del LEL puede ser una tarea simple de poco esfuerzo.

Por otro lado, es frecuente que las organizaciones se interesen en este modelo como un documento independiente del software, por su simplicidad y riqueza, con el fin de emplearlo, por ejemplo, como un instrumento de capacitación a nuevos integrantes en dicha organización.

Durante la comprensión del UdeD actual, es frecuente la detección de algunos defectos en el LEL, los cuales deben corregirse al ser descubiertos. La incompletitud no debe considerarse un defecto en sí mismo, excepto cuando la

omisión de información pueda llevar a una mala interpretación del significado de un símbolo del LEL. Durante la comprensión del UdeD futuro, existe una posibilidad bastante menor de encontrar algún defecto, aquí nuevamente ocurre un mantenimiento correctivo, el cual puede realizarse en cualquier momento del ciclo de vida del software pero con una muy baja frecuencia.

Es recomendable una vez instalado el sistema de software parcial o totalmente, según el modelo de proceso de software seguido, realizar un mantenimiento evolutivo si la incorporación del nuevo software provocó un cambio en el vocabulario del UdeD.

¿Por qué mantener los EA?

Al igual que en el caso del LEL, puede existir el riesgo de defectos en los EA. Entonces el mantenimiento correctivo deberá realizarse siempre y cuando se haya determinado la necesidad de mantener la evolución del UdeD y por ende los EA, o cuando este defecto haya acarreado un defecto en los EF y la evaluación de la corrección en los EF se facilite a través de la corrección previa en los EA.

En un proyecto de gran envergadura o donde el UdeD es complejo, y es necesario implementar ya un cambio en el negocio, puede ser difícil imaginar su incidencia en el futuro dada la situación presente. En un caso así, la tarea de analizar los cambios que ocurren en el contexto actual puede facilitarse estudiando los EA y modificándolos, para luego trasladar los cambios a los EF. En general, el tipo de mantenimiento que se trata aquí es evolutivo y eventualmente adaptativo.

Por otro lado, el mantenimiento de los EA, evita depender de la estabilidad del equipo de ingenieros que realizan o realizaron el proceso de IR en el proyecto. En organizaciones de gran rotación del personal de desarrollo entre proyectos, contar con documentación que describa los procesos del negocio actuales facilita el ingreso de cualquier persona al equipo de desarrollo en cualquier momento del proyecto. Esta misma razón también es válida para el mantenimiento del LEL.

En el peor de los casos, los EA deben mantenerse hasta la implantación del software, donde ellos se vuelven un documento histórico, sólo para consulta o para el rastreo de los requisitos hacia sus orígenes.

¿Por qué mantener los EF?

El conjunto de EF es el “núcleo central” de SDRES, pues reflejan todos los requisitos del software, son fáciles de interpretar por los clientes y usuarios, de manera que cualquier cambio a realizarse durante el proceso de desarrollo puede ser fácilmente validado con ellos, y manifiestan el camino a seguir durante el proceso de desarrollo para todo el equipo de trabajo. Adicionalmente, como ya se mencionó en la sección anterior, los EF sirven de

ancla para la pre y post rastreabilidad de los requisitos, y por ende es indispensable su mantenimiento. Luego, esto facilita el traslado de los cambios a los artefactos construidos en las etapas subsiguientes del proceso de desarrollo.

Cualquier tipo de cambio a realizarse debe quedar reflejado en este conjunto de escenarios, pues su utilidad se extiende además a la fase de operación donde ellos han evolucionado convirtiéndose en los EA. Durante esta fase, al realizarse una actualización importante en el software, debería tomarse a los nuevos EA como base para construir los EF que representarían las situaciones proyectadas según los nuevos cambios, reiniciándose así el ciclo de evolución de los escenarios.

¿Por qué mantener el SRS?

Esto depende en gran medida de la relación cliente-desarrollador. En caso de usarse el SRS como parte del contrato entre las partes, los cambios en los requisitos deberán reflejarse en ajustes al contrato.

En SDRES, el SRS puede mantenerse actualizado derivando requisitos de los EF una vez actualizados éstos, donde es más fácil incluir los cambios, en lugar de incorporarlos directamente en el SRS.

3.4. Cómo aplicar y adaptar SDRES

Según el proceso de desarrollo de software que se esté utilizando, las etapas de SDRES podrán adaptarse para ser parte de dicho proceso. Se evalúan a continuación los modelos de proceso descriptos en la sección 2.3 y la aplicabilidad de SDRES a ellos.

En un proyecto que sigue un modelo de cascada, SDRES se desarrolla como la primera fase del proceso de desarrollo, realizando sus cuatro etapas consecutivamente: se comienza conociendo el vocabulario del UdeD, luego se comprenden las actividades del negocio, luego, se proyectan las actividades del negocio considerando el nuevo sistema de software y, por último, se establecen los requisitos del software del contexto, expresándolos en términos del vocabulario del UdeD.

Según lo expresado en la sub-sección 2.2.3, el proceso de definición de requisitos se lleva a cabo en forma iterativa, luego SDRES en su segunda etapa, retrocede a la primera refinando el LEL, pues es común que durante la comprensión del UdeD se describa uno o más términos nuevos o se mejore la definición de los ya existentes, y en la tercera etapa pueden ocurrir también ajustes menores al LEL menos frecuentes y actualizaciones a los EA al requerir a veces mayores precisiones para proyectar nuevas situaciones que suponen el uso del nuevo sistema de software.

En un proyecto que sigue un desarrollo iterativo incremental, aplicando

SDRES se recomienda construir inicialmente un LEL de todo el vocabulario del UdeD que se va a estudiar. Una propuesta sería continuar luego dividiendo el proyecto en incrementos para construir los EA de las partes que correspondan a ese incremento, y sobre esa base, construir los EF de ese incremento. La dificultad de esta propuesta radica en cómo dividir en partes el problema que aún no se conoce. Una solución puede ser la construcción inicial de algunos pocos escenarios actuales denominados "generales" que no entran en detalles del problema, cuyo único fin es poder realizar la división (esta propuesta se discutirá en el capítulo 6). Una propuesta más viable es la construcción de los EA del UdeD y, una vez conocido el problema, realizar la partición del mismo, e ir construyendo en cada incremento los EF afectados a él. En cada ciclo, parte de los EF pasan a transformarse en EA debido a la implantación parcial del software, dado que ellos pasan a representar el nuevo UdeD actual (ver sub-sección 3.3.2).

El modelo de prototipos, tanto desechable como evolutivo, se centra en la creación de prototipos para elicitar y modelar, mientras que SDRES en cambio se basa en los escenarios como técnica de elicitación y modelado, sugiriendo la construcción de prototipos para validar escenarios. En el modelo de especificación operacional sucede algo similar, ya que se basa en la construcción de un modelo ejecutable para modelar y validar los requisitos junto con factores de diseño técnico. La elicitación es un paso previo a la construcción de la especificación operacional. Luego, SDRES no es aplicable en estos dos modelos de proceso.

El modelo de transformación formal parte de un documento de definición de requisitos que puede perfectamente corresponder con los EF o con el SRS construidos a partir de SDRES. Por ejemplo, Mauco & George [Mauco 00] [Mauco 01] generan especificaciones formales escritas en el lenguaje de especificación RAISE a partir del LEL y de escenarios. Luego, en este proceso, se realizarían sólo las tres primeras etapas de SDRES.

El modelo espiral obtiene como producto del segundo ciclo los requisitos del software validados, este ciclo puede llevarse a cabo siguiendo SDRES que incluye la elicitación, el modelado, la verificación y la validación de requisitos de software. La evaluación de alternativas en este ciclo se manifiesta en SDRES mediante la construcción de conjuntos alternativos de EF. El prototipo de esta iteración corresponde con el prototipo que se construye para validar los EF.

El modelo basado en la reutilización de componentes incluye una primera fase de especificación de requisitos, que puede realizarse con SDRES, y en su tercera etapa de adaptación de requisitos a componentes disponibles, cabe aquí la evolución de los EF y del SRS. La evolución de los requisitos no es tratada por este modelo, por lo que se vuelve a empezar el proceso especificando los requisitos y evaluando componentes que cubran los nuevos requisitos.

En la Tabla 3-1 se presenta una síntesis de dónde y cómo aplicar SDRES en cada modelo de proceso de software si corresponde.

Modelo de Proceso	Aplicación de SDRES
Modelo de Cascada	En la 1ª fase Análisis de Requerimientos: se aplica la estrategia completa en sus cuatros pasos secuencialmente.
Modelo de Prototipado	No aplicable.
Modelo de Especificación Operacional	No aplicable.
Modelo de Transformación Formal	En la 1ª fase Definición de Requisitos: se realiza la estrategia completa o sólo las tres primeras etapas según se comience por la formalización desde el SRS o desde los EF.
Desarrollo Iterativo e Incremental	1° se construyen el LEL y los EA. 2° se divide en iteraciones en base a los EA. 3° se construyen los EF en cada iteración.
Modelo Espiral	En la 2° iteración de la espiral, donde se producen los requisitos del software validados: se aplica la estrategia completa.
Modelo basado en Reutilización	En la 1ª etapa Especificación de Requisitos: se aplica la estrategia completa. En la 3ª etapa Adaptación de Requisitos a Componentes: se modifican los EF y el SRS.

Tabla 3-1. Aplicación de SDRES en los Modelos de Proceso

Como se mencionó en la sección anterior, no todos los modelos requieren ser actualizados, pero cabe mencionar además que no todos los modelos requieren ser construidos en esta estrategia. Dado lo mencionado para el CORB, el conjunto de EF es el modelo pilar de esta estrategia por contener los requisitos del software. Como SDRES favorece la comunicación entre los involucrados haciendo un uso intensivo del vocabulario del UdeD, esto involucra entonces la construcción del LEL, y además la estrategia brinda heurísticas para detectar situaciones utilizando el LEL. Es por ello que se recomienda siempre la construcción del LEL. Aunque se podría aducir, que en un proyecto pequeño de baja complejidad esta actividad sería innecesaria, pero en tal caso es probable que cualquier estrategia ad-hoc pudiera realizarse sin presentar grandes inconvenientes.

Luego en lo que sigue se discutirá la conveniencia o no de construir los restantes artefactos: EA y SRS. Para ello se deben evaluar ciertos factores de la organización y del proyecto en particular, tales como el conocimiento previo que el equipo de desarrollo posea acerca del UdeD bajo estudio, la envergadura del proyecto y la complejidad del UdeD. A continuación se dan guías de selección de los modelos a construir según cada caso.

Según el conocimiento previo del UdeD:

Si el equipo de desarrollo es parte de la organización cliente y conoce el UdeD, es poco necesario construir los EA, salvo que sea necesario resaltar los problemas actuales o justificar los cambios a introducir a través del nuevo software, o que intervengan además algunos de los otros factores mencionados.

El peor problema frente a la decisión de construir los EA es cuando los desarrolladores "creen que conocen" por experiencia el UdeD, y entonces desestiman la producción de este artefacto. La construcción del LEL ayudará a ratificar o rectificar las creencias de los desarrolladores, y a partir de ello decidir sobre la construcción de los EA.

Un ejemplo de un caso donde se consideró innecesaria la construcción del LEL y de los EA fue para el "Sistema de producción y exportación de artículos de cuero"[53], pues era un contexto conocido por los desarrolladores y con usuarios experimentados. Pero posteriormente durante el avance del desarrollo surgieron varios inconvenientes debido a la sorpresiva y sucesiva rotación de los usuarios y los cambios constantes en los procesos del negocio al mudarse la fábrica a un predio más amplio y moderno.

Según la rotación del equipo de desarrollo:

Cuando hay una gran rotación de los integrantes del equipo de desarrollo, entre distintos proyectos, es altamente recomendable construir los EA, para servir de reservorio organizado del conocimiento adquirido, facilitando el entrenamiento de nuevos integrantes al proyecto.

Según la envergadura del proyecto[54]:

En un proyecto pequeño a mediano es probable que no justifique por los tiempos y recursos requeridos la construcción de los EA, salvo que intervenga el factor complejidad del UdeD. Mientras que en proyectos medianos a grandes, es absolutamente justificable y necesaria la realización de las cuatro etapas completas de SDRES, pues por un lado, se trabaja con un equipo de desarrolladores con tareas bien definidas, donde estos dos modelos sirven de fuente de conocimiento para todo el grupo, y donde en general es muy difícil mantener todo el conocimiento en la mente de los ingenieros; y por otro lado, en estos casos es muy importante el rastreo de los requisitos a sus orígenes,

[53] "Sistema de producción y exportación de artículos de cuero" fue desarrollado por Gladys Kaplan y la autora entre 2001 y 2006 en una industria argentina. El sistema incluyó módulos de control de pedidos de compra, gestión de plantillas de artículos a producir y semielaborados, mantenimiento de stocks de productos, semielaborados y materias primas, control de producción y gestión de exportaciones e importaciones.

[54] Para establecer la envergadura de un proyecto se pueden considerar los siguientes atributos: cantidad de usuarios, tamaño del equipo de desarrollo, duración del proyecto, presupuesto asignado y el tamaño estimado del software (medido en cantidad de componentes, cantidad de requisitos, TLOC).

poder comparar lo actual frente a lo propuesto y justificar dichas propuestas.

Por otro lado, en un proyecto de gran envergadura se suele generar un SRS como parte del contrato.

Según la complejidad del UdeD[55]:

En el caso de un UdeD que se podría considerar simple por la baja complejidad en los procesos del negocio, en general es poco necesaria la construcción de los EA. A mayor complejidad en las actividades que se desarrollan en dicho contexto, es casi inevitable la construcción de los EA.

En caso de un UdeD de gran complejidad, es conveniente generar un SRS para facilitar la gestión de los requisitos permitiendo un control individual de los mismos.

Según las características de la organización cliente:

Se considerará en este rubro las siguientes características: la novedad del negocio, estándares internos o externos exigidos, la frecuencia de cambios en los procesos del negocio, la rotación del personal y el tipo de cliente.

Obviamente, en el caso de un negocio nuevo, no tiene sentido hablar de la construcción de EA, donde a partir del LEL podrá proyectarse un conjunto de EF, avizorando varias posibles situaciones alternativas.

En el caso de estándares de especificación que sean exigidos contractualmente o no, se deberá construir el SRS de acuerdo a dichos estándares, basándose en los EF previamente generados.

Cuando hay una alta rotación de los usuarios, es altamente recomendable la construcción de los EA, para evitar malos entendidos con el nuevo personal que dificulten el avance normal del proyecto. Adicionalmente, podrán ser empleados como instrumentos para la capacitación del nuevo personal de la organización cliente.

En el caso de un UdeD que sufre constantes cambios, se deberá evaluar conforme a los restantes factores la conveniencia o no de construir y mantener los EA. De establecerse el beneficio de construirlos como base para generar los EF, podrán luego ser descartados con el fin de evitar una constante actualización de los mismos.

Otra consideración es si el proyecto involucra el desarrollo para un cliente en particular o para un mercado de potenciales clientes. En el primer caso se realiza SDRES en todas sus fases o no según los restantes factores,

[55] La complejidad implica el grado de dificultad para imaginar el futuro haciendo una abstracción mental de los problemas actuales o mejoras deseadas respecto a la realidad existente.

mientras que para el segundo caso, la realización de la segunda fase de SDRES depende de otras consideraciones, pues en principio se puede presuponer no construir los EA dado que no hay situaciones actuales para una organización específica. Pero esta segunda fase puede realizarse para comprender el estado actual de la oferta en el mercado de diferentes sistemas de software para una aplicación dada, construyendo entonces diferentes conjuntos de EA que los representen, y a partir de ellos evaluar la propuesta de software a desarrollar. Bajo esta premisa, es viable entonces también realizar la primera etapa que describa los términos utilizados en el mercado, resaltando los posibles sinónimos y homónimos (ver sección 4.3).

3.5. Ámbito de aplicación de la estrategia

Es fundamental para la comunicación que los involucrados en un proyecto compartan el mismo lenguaje. Para ello existen tres posibilidades: i) que los ingenieros de software adquieran conocimiento sobre el léxico de los clientes y usuarios, ii) que a la inversa los clientes y usuarios se acerquen al vocabulario técnico de los ingenieros, o iii) que ambas partes comprendan y utilicen un tercer léxico.

La estrategia que se plantea en esta tesis se basa en la primera alternativa, que en la mayoría de los casos es viable de poner en práctica, pues el esfuerzo de aprendizaje recae en el grupo desarrollador. La segunda alternativa tiene inconvenientes con frecuencia insuperables: no siempre se cuenta con clientes y usuarios proclives a aprender, con tiempo disponible para ser capacitados o con interés hacia la informática. Sin embargo, existen contextos en los cuales se requiere que los ingenieros de software realicen el estudio de una disciplina o una especialización exhaustiva en un tema, donde también se pueden presentar dificultades similares: largos períodos de aprendizaje, o áreas de escaso interés o de alta complejidad de aprendizaje. Para estos casos no es aplicable esta estrategia que se centra en el uso de modelos escritos usando el vocabulario del UdeD. Mientras que la tercera opción es irreal en este contexto.

Dados los casos de estudio y casos reales desarrollados siguiendo SDRES, se puede decir que es una buena estrategia cuando existen interacciones entre humanos y máquinas de cualquier índole, y cuando la naturaleza del problema es razonablemente simple de comprender por un profesional de informática. También resulta adecuada cuando el alcance del problema no es claro, los escenarios especialmente ayudan a delimitarlo.

Sirve para estudiar problemas de complejidad estructural, es decir, problemas compuestos por una gran cantidad de elementos y lazos, pero no es aplicable a problemas de complejidad intrínseca de sus elementos. Esto último se debe a la dificultad de transmisión de información desde el UdeD a los desarrolladores o la absorción de dicha información por parte de los desarrolladores.

Además, puede considerarse una estrategia adecuada para sistemas

donde deben estudiarse las características sociales del trabajo actual para poder considerar las características técnicas del nuevo sistema, tal es el caso de los denominados sistemas socio-técnicos[56].

3.6. Versionado de los modelos

Los modelos en LN suelen implementarse hoy en día con sistemas de hipertexto que proveen la facilidad de vincular párrafos dentro del documento de soporte del modelo, así como párrafos externos al mismo en otros documentos hipertextuales.

Estos vínculos de carácter semántico se establecen entre elementos del modelo o de distintos modelos, ya sea por reconocimiento automático de texto, por ayuda en el contexto o en forma manual. En el caso del LEL, un ejemplo de vínculo interno es una referencia (mención) a un símbolo en la descripción de otro símbolo; dicha conexión entre la mención al símbolo y la definición del mismo se implementa con un hipervínculo. Un ejemplo similar es la referencia a un EA en la descripción de un episodio de otro EA. En ambos casos, es probable un reconocimiento automático. Un vínculo entre un EA y un símbolo del LEL, o entre un EF y un requisito son casos de un vínculo externo, donde en el primer caso un reconocimiento automático es factible y en el segundo un método manual es más probable.

Es entonces, que los modelos propuestos por SDRES se han vinculado fácilmente entre ellos, aunque debe también considerarse la vinculación de éstos con otros modelos no hipertextuales o no escritos en LN, pues por razones de trazabilidad de los requisitos se desea establecer dependencias con modelos fuera del CORB, como por ejemplo modelos de diseño, modelos de implementación, componentes de software, o fuentes de información. Estos nexos semánticos externos deben ser generados aunque basados en otros mecanismos, ya no tan simples como los hipervínculos. Estos hipervínculos no impiden el uso de otra estrategia para la trazabilidad, ni tampoco para resolver la problemática del versionado que requiere una estrategia más compleja.

Específicamente en el área de gestión de configuración, existen otras conexiones que relacionan distintas versiones del mismo modelo, las cuales a su vez pueden relacionarse cada una con distintas versiones de otros modelos. Estas relaciones manifiestan los cambios de estado de un modelo debido al pedido de cambios en requisitos. Estos cambios simultáneamente pueden desencadenar cambios en otros modelos relacionados semánticamente. Luego, a través de los nexos entre versiones se obtiene la historia de los cambios ocurridos en un modelo o entre modelos. La facilidad de manejo de vínculos hipertextuales podría extenderse para la gestión del versionado, pero esto evitaría el manejo de información asociada al versionado, como el motivo y momento del cambio. Además, la navegación entre el LEL, los EA, los EF y el SRS es mucho más predecible y responde a patrones de pensamiento más conocidos que los seguimientos de trazas en problemas de versionado. Esto es

[56] Un sistema socio-técnico es una interrelación compleja entre personas y tecnología, incluyendo hardware, software, datos, ambientes físicos, personas, procedimientos, leyes y regulaciones [Maté 05].

debido a que, por ejemplo, la consulta sobre una duda en un EF casi siempre involucra la revisión de la situación homóloga actual (EA) o un símbolo del LEL.

Tipo de Nexo	Motivo	Implementación	Alcance
Natural	Representación de los modelos	Hipervínculo o puntero en un modelo de datos de red	Intra CORB[57]
Semántico	Trazabilidad de requisitos	Modelo de datos relacional u objeto relacional con vistas y consultas	Intra CORB
		Mecanismo dependiente de los modelos a relacionar	Extra CORB
de Versionado	Reflejar la gestión de cambios	Modelo de datos relacional con vistas y consultas	Intra CORB
		Mecanismo dependiente de los modelos a relacionar	Extra CORB

Tabla. 3-2. Tipos de Nexos entre Modelos

En resumen, como se muestra en la Tabla 3-2, mediante vínculos semánticos se puede recorrer los elementos de un modelo en LN y a partir de ellos recorrer otros modelos en LN, apoyando la integración de conocimiento. Mediante nexos de versionado, a partir de la versión actual de un modelo, se consultan los cambios solicitados hasta reconstruir la versión anterior o se busca un cambio específico, para una revisión o regreso al pasado. Dependiendo de la implementación de nexos, es factible lograr una gran variedad de consultas referidas al versionado. Incluso mediante estos nexos se puede consultar a partir de la versión de un modelo la versión correspondiente de otros modelos. Es decir, estos nexos potencian los vínculos semánticos permitiendo una búsqueda o recorrido correcto entre los elementos apropiados de un modelo y entre los modelos apropiados en un momento dado.

El CORB (ver Figura 3-8) crece con cada nuevo modelo de requisitos que se incorpora durante la IR y se refina con cada nueva versión que se requiera de dichos modelos durante todo el proceso de desarrollo del software. En cada hito establecido del proyecto, el CORB es una fotografía de lo que está "operativo" para su distribución a los clientes y usuarios, es decir la última versión de los modelos de requisitos. Una copia de este baseline queda para continuar siendo trabajada por avance del proceso de desarrollo en sí mismo y/o por cambios solicitados, hasta que se decida emitir una nueva versión de estos modelos. La generación de una versión no ocurre por la llegada del proyecto a un hito, sino por el volumen de cambios registrados desde la última

[57] Intra CORB: incluye las relaciones Inter Modelo y Extra Modelo dentro del CORB.

versión del modelo. En la Figura 3-8, se observan vínculos intra CORB, las líneas simples son vínculos naturales dentro de un modelo, las flechas de punto representan vínculos naturales entre modelos y las flechas sólidas de doble sentido son nexos entre versiones.

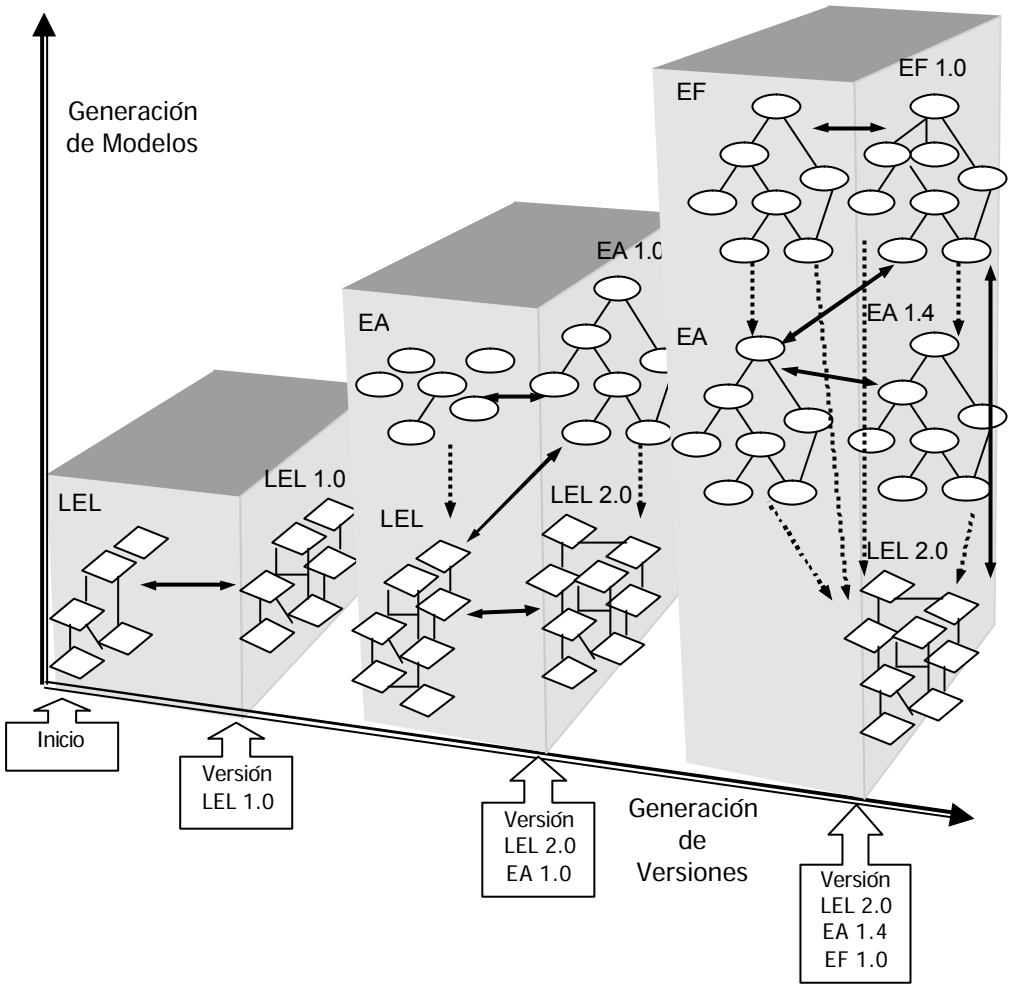
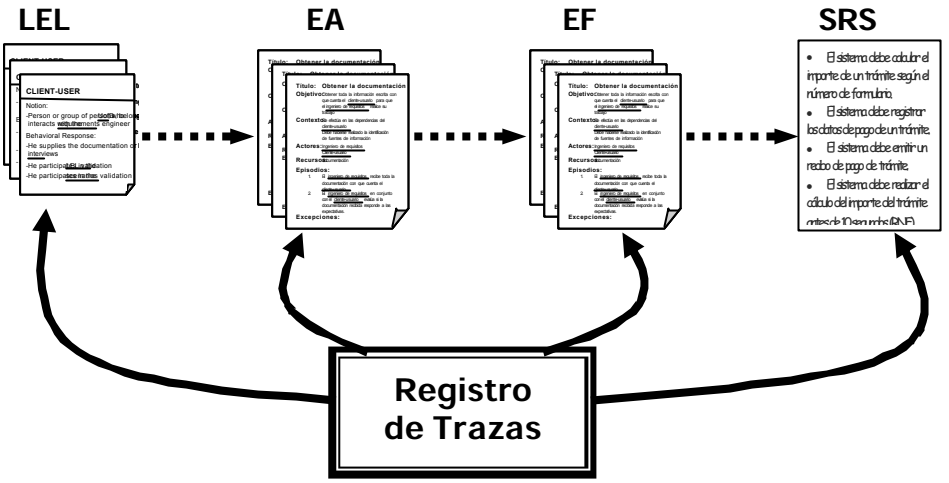


Figura 3-8. Nexos en el CORB[58]

En el campo del versionado, existe una gran variedad de estrategias de versionado [Sommerville 02], que abarcan desde la forma de generación de versiones hasta el mantenimiento y control de cada cambio. Por un lado, se debe decidir el nivel de especificidad de las versiones: por cada elemento individual del modelo, por grupos de elementos, por el modelo íntegro, e incluso por un conjunto consistente y cohesivo de modelos. Es decir, se debe determinar el tipo de ítem a versionar. La estrategia debe identificar la frecuencia de emisión de una versión: desde una nueva versión por cada cambio efectuado, o el acumulado de sucesivos cambios hasta la decisión de una nueva versión del ítem. La estrategia debe determinar cómo guardar cada cambio realizado sobre cada elemento, y para ello existen varias formas de almacenamiento de cambios:

[58] Por razones de espacio, el modelo SRS no se incluyó en el gráfico.

i) versión vieja + Δ cambio;
ii) versión vieja y versión nueva;
iii) versión nueva - Δ cambio.



Ref.LELv2.0, Símbolo 46, Noción 2.	Ref.LELv2.0, Símbolo 4.	18-10-2005	Alta de vínculo al nuevo símbolo por Pedido nro. 3894

Ref. EA v1.4, Escenario 18, Episodio 5.	---	20-10-2005	Mejora en Sector Expedición tarea actual embalado con lápiz óptico.

Figura 3-9. Registro de Trazas para Versionado

El obstáculo de la primera opción es la dificultad en la legibilidad del estado actual pero aporta una buena visibilidad de los cambios. La segunda opción tiene buena legibilidad de ambos estados pero sin reconocer los pasos transitados. La tercera opción presenta el estado actual sin problemas y también buena visibilidad de los cambios, yendo hacia atrás pero sin saber hasta donde retroceder. Debe tenerse en cuenta también que llevar una historia de cambios para reconstruir versiones hacia atrás o hacia delante es una opción costosa por la complejidad que ello involucra. La segunda opción puede mejorarse llevando una historia simple de cambios, sin requerir a partir de ella reconstrucciones de versiones. Es decir, que la traza permite reconocer el contexto y los motivos del cambio sin la complejidad del registro de todos los detalles que involucra el mismo. Debe analizarse el par beneficio-costo de cada opción para cada proyecto en particular.

Para el caso de modelos en LN, la segunda opción de bajo costo comparativo es recomendable, y las trazas se pueden manejar en forma simple por apilamiento. Una traza puede informar la causa de un cambio, quién y cuándo se realizó y los efectos producidos: lista de símbolos, de EA, de EF y/o de requisitos afectados. En el Anexo E, se presenta una propuesta de trazas para versionado de los modelos de SDRES. Este modelo de trazas, ilustrado en la Figura 3-9, está superpuesto a los modelos en LN, de manera tal que si

se elimina el modelo de trazas, los modelos en LN no pierden información, ya que están auto-contenidos en su propia red de dependencias (hipervínculos).

3.7. Resumen

Se ha presentado en este capítulo una estrategia en la Ingeniería de Requisitos basada en modelos en lenguaje natural, que enfatiza la adquisición de conocimiento sobre el contexto organizacional donde operará el sistema de software. Esta estrategia, aunque expuesta siguiendo un modelo en cascada, es adaptable a diversos procesos de software, y es utilizable en una amplia variada de problemas. Adicionalmente, se dan pautas de cómo adaptar la estrategia a distintos proyectos y organizaciones.

La estrategia se basa en la "elicitación dirigida por modelos", de manera tal que se apoya en un modelo auxiliar para aquella información obtenida espontáneamente y que no es transferible al modelo en curso de construcción. Ese modelo auxiliar, denominado Ficha de Información Anticipada, evita la pérdida de información capturada en forma adelantada al modelo receptor de la misma.

Para la gestión de los requisitos, se ha presentado un modelo simple de trazas que facilita el versionado de los modelos construidos y mediante hipervínculos se facilita una navegación entre los modelos del CORB, que junto con un modelo de datos relacional u objeto relacional permite establecer la rastreabilidad de los requisitos dentro del CORB.

Parte del material incluido en este capítulo ha sido publicado en [Leite 04b].

Capítulo 4
Léxico Extendido del Lenguaje

When **Confucius** was asked how to govern, he said, **«by putting names right** and added **«If names are not right, words are misused. When words are misused, affairs go wrong. When affairs go wrong, courtesy and music droop. When courtesy and music droop, law and justice fail. So a gentleman must be ready to put names in speech, to put words into deed. A gentleman is nowise careless of words».**

"The Sayings of Confucius" en "The Harvard Classics and Harvard Classics Shelf of Fiction", 1909-1917

4.1. Introducción

Si los involucrados en un proyecto comparten el mismo lenguaje entonces la comunicación entre ellos mejora [Leite 89], y una alternativa productiva se presenta cuando los ingenieros de software usan tanta jerga de los clientes y usuarios como sea posible [Malinowski Rubio 01]. Como resultado aumentará el compromiso de los clientes y usuarios con el proyecto [Bubenko 93] [Macaulay 93]. Además, en el proceso de IR, la validación de los diferentes productos requiere una interacción fuerte con los clientes y usuarios [Loucopoulos 95]; esta actividad se facilita usando un vocabulario común a todos los involucrados y que corresponda al dominio de la aplicación. Este vocabulario del dominio de la aplicación no sólo debe usarse durante las interacciones entre los involucrados sino también en los modelos y documentación del software que los desarrolladores produzcan.

La creación del LEL como el primer paso en la definición de requisitos apunta a entender el vocabulario del dominio de la aplicación. Por b tanto, la manera de comunicarse con los clientes y usuarios, y de entender el problema permite tener un obstáculo menos: el vocabulario.

Varios autores, como se expuso en la sub-sección 2.7.2, sugieren el uso de un glosario que contenga el vocabulario del dominio de la aplicación. El LEL es un glosario con roles y estructura más amplia que lo usual. No es sólo un glosario descriptivo sino también prescriptivo, porque introduce definiciones más precisas que evitan la vaguedad y ambigüedad de las definiciones descriptivas. Pues el LEL involucra la denotación y la connotación de cada símbolo considerado como una palabra o frase pertinente al dominio de la aplicación. Los términos del léxico son las designaciones en la terminología de Jackson [Jackson 95] y representan los hechos del mundo real expresados en su propio lenguaje. El propósito de crear este léxico no sólo es permitir una buena comunicación y acuerdo entre los involucrados, sino también ser el puntapié inicial en el proceso de construcción de escenarios [Leite 00], ayudar en la descripción de los mismos y mejorar el proceso de validación de los escenarios cuando se sigue el enfoque SDRES.

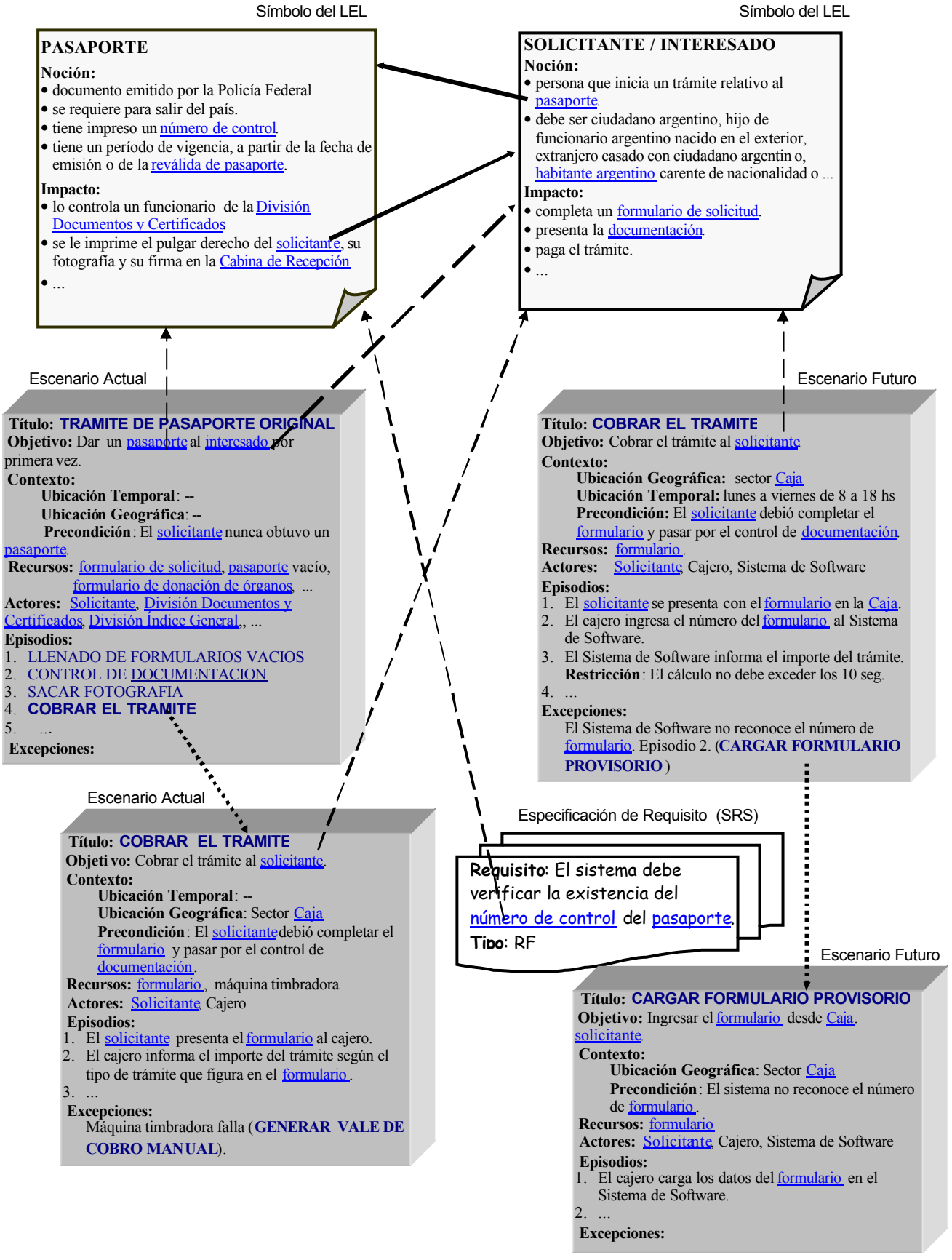


Figura 4-1. Ejemplo de la vista hipertexto entre el LEL, Escenarios y SRS

Tanto los escenarios como el SRS se describen usando los símbolos del LEL, esta característica establece un vínculo natural entre estas tres estructuras de representación. Hay también vínculos naturales dentro del léxico dado que en la descripción de un término se hace uso de otros términos.

La Figura 4-1[59] muestra un ejemplo que ilustra ambos tipos de vínculos, con flechas llenas vínculos dentro del LEL y con flechas guionadas vínculos entre el LEL y los otros modelos. Adicionalmente, se presentan en la figura dos tipos de vínculo entre escenarios con flechas punteadas, uno que se establece debido a la relación jerárquica basada en el concepto de escenario-sub-escenario, y el otro como consecuencia de la relación de excepción entre un escenario y otro escenario que trata la excepción (ver capítulo 5).

Si se sigue otra estrategia en la IR, el LEL también podría usarse como un ancla para las próximas actividades de IR y las subsecuentes fases de desarrollo de software. Por ejemplo, Colin Potts [Potts 99] hace notar sobre la utilidad de un léxico de verbos para identificar objetivos y tareas durante el paso de identificación y refinamiento de metas de la estrategia ScenIC. Whitenack en [Whitenack 94] propone la definición de objetos del dominio a partir de un glosario creado previamente. Breitman & Leite [Breitman 03] proponen la creación de un LEL para usarlo posteriormente en la construcción de la ontología de una aplicación.

Como el vocabulario es un ancla [Leite 99], se puede hacer rastreo a través de las palabras y frases que llevan a diferentes designaciones. Además, el léxico permite el entrenamiento de miembros del equipo de desarrolladores que son incorporados en fases posteriores durante el desarrollo del software.

En resumen, los problemas que atiende el LEL son básicamente problemas de comunicación y problemas de comprensión, siendo entonces sus objetivos y sub-objetivos:

- Comprender el lenguaje del dominio de la aplicación:
  - asegurar una buena comunicación entre los involucrados;
  - facilitar la validación con los clientes y usuarios de cualquier documento de requisitos escrito en LN;
  - preservar el mismo vocabulario durante el ciclo de vida del software;
  - facilitar la comprensión del UdeD en las actividades de la IR.

- Ser un ancla para todas las fases del proceso de desarrollo del software:
  - ser un punto de partida para las siguientes actividades de la IR;
  - permitir la búsqueda de información acerca del UdeD en un modo no estructurado, es decir, convertirse en un repositorio de conocimiento sobre el UdeD;

[59] Este ejemplo fue extraído del caso de estudio "Sistema Nacional para la obtención de Pasaportes" desarrollado en 1996 durante el proyecto de investigación Uso de Escenarios en el Desarrollo de Software en la Universidad de Belgrano.

- ⇨ facilitar el entrenamiento de nuevos miembros del equipo de desarrollo sobre la terminología en uso;
- ⇨ ser una fuente para la convención de nombres en el diseño y codificación;
- ⇨ ayudar durante la producción de la documentación de usuarios;
- ⇨ ser un instrumento simple de rastreabilidad.

4.2. El modelo del LEL

El Léxico Extendido del Lenguaje [Leite 93] [Leite 89] es un meta-modelo diseñado para ayudar en la elicitación y representación del lenguaje usado en el dominio de la aplicación. El proceso de creación se ancla en la idea de entender sólo el lenguaje del dominio de la aplicación, como un primer paso para mejorar la comprensión sobre el UdeD. Para apreciar esta idea cabalmente se debe notar que no se está desestimando la comprensión del problema, sino que se está solamente definiendo la precedencia del lenguaje sobre el problema mismo durante la creación del LEL.

LEL: representación de los símbolos en el lenguaje del dominio de la aplicación.
Sintaxis:
<LEL> ⇨ $\{<\text{Símbolo}>\}_1^N$

Símbolo: entrada del léxico que tiene un significado especial en el dominio de la aplicación.
Sintaxis:
<Símbolo> ⇨ $\{<\text{Nombre}>\}_1^N$ + <Noción> + <Impacto>

Nombre: identificación del símbolo. Más de uno indica la presencia de sinónimos.
Sintaxis:
<Nombre> ⇨ <Palabra> | <Frase> | <Acrónimo>

Noción: denotación del símbolo. Debe ser expresada usando referencias a otros símbolos y usando el vocabulario mínimo.
Sintaxis:
<Noción> ⇨ Noción: + $\{<\text{Oración}>\}_1^N$

Impacto: connotación del símbolo. Debe ser expresado usando referencias a otros símbolos y usando el vocabulario mínimo.
Sintaxis:
<Impacto> ⇨ Impacto: + $\{<\text{Oración}>\}_1^N$

donde
<Oración> está compuesta solamente por Símbolos y No Símbolos, éstos últimos pertenecientes al vocabulario mínimo;
<Palabra>, <Frase> o <Acrónimo> tienen el significado común.

+ significa composición, {x} significa cero o más ocurrencias de x, | representa **or**

Figura 4-2. El Modelo del Léxico Extendido del Lenguaje

El LEL es en sí mismo un glosario con roles y estructura diferentes al usual, y contiene hipervínculos entre sus entradas. Está compuesto por un conjunto de símbolos que son, en general, palabras o frases peculiares y las más usadas en el dominio de la aplicación. Una sigla también puede ser el nombre de un término cuando se usa en el dominio de la aplicación. Cada

símbolo se identifica por un nombre o nombres (en caso de sinónimos) y tiene dos tipos de descripciones; este formato particular representa la diferencia con otros glosarios. El primer tipo, llamado Noción, es el usual pues describe la denotación[60] de la palabra o frase, es decir, define "lo que el símbolo es". El segundo, denominado Impacto, describe la connotación de la palabra o frase, es decir, describe "cómo el símbolo actúa en el dominio de la aplicación"; esta descripción no está presente normalmente y enriquece el conocimiento sobre el símbolo y el contexto.

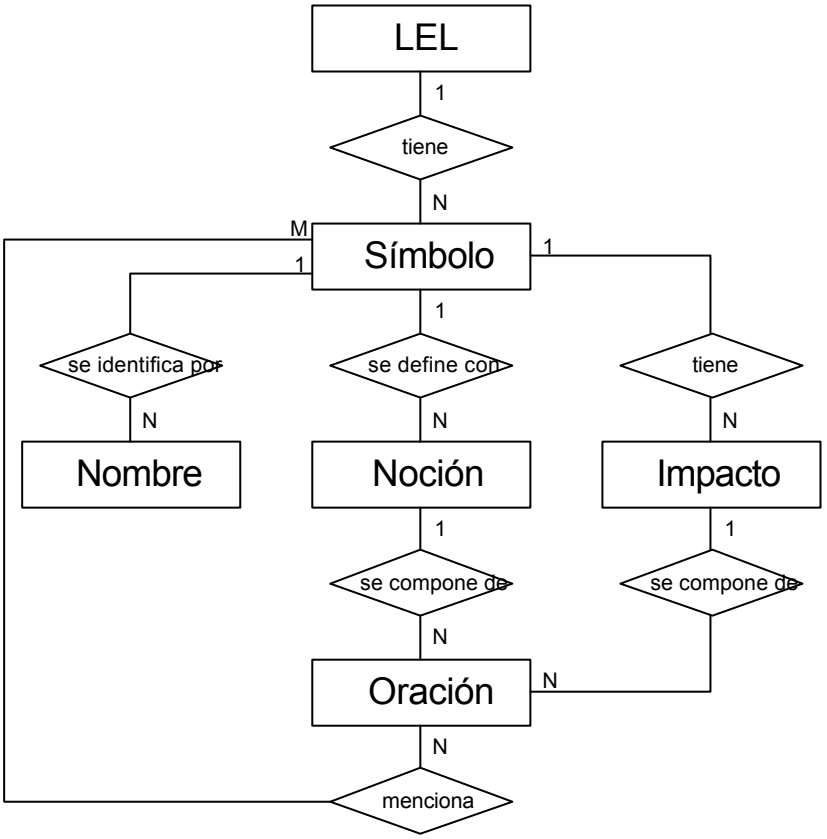


Figura 4-3. Diagrama de Entidad-Relación del Modelo del LEL

Como se ha mencionado en la sub-sección 2.7.1, un diccionario proporciona varios significados de un símbolo mientras que el LEL tiene un espectro más restringido: un dominio específico, y los sinónimos del símbolo se usan para reforzar el significado básico. Es decir, todos los nombres asignados a un símbolo comparten el mismo exacto significado, y se ignoran en forma intencional aquellos significados provenientes de otros contextos.

La Figura 4-2 presenta el Modelo del Léxico Extendido del Lenguaje, el cual muestra sus entidades junto con su sintaxis, y en la Figura 4-1 se

[60] Cuando la denotación del término tiene varios significados, entonces cada significado indica la presencia de más de una entrada en el léxico (ver sub-sección 4.3.3: Tratamiento de Homónimos). Cuando la denotación coincide con el significado, el término está libre de contexto. Pero, en muchos casos, la denotación y el significado no coinciden; este es el caso de un término usado en más de un contexto específico [Parianou 01].

presentan dos ejemplos de símbolos, donde los términos subrayados representan vínculos a otras entradas del LEL.

Dos principios [Leite 90] rigen la descripción de los símbolos. El principio de circularidad, también denominado "principio de clausura", declara la maximización del uso de símbolos en la descripción de otros símbolos y el principio del vocabulario mínimo establece la minimización del uso de términos que son externos al léxico. Estos términos externos deben pertenecer a un pequeño subconjunto de un diccionario predefinido del LN que no contiene ningún símbolo del LEL. Por ejemplo, en Inglés las palabras en el Collins Dictionary's COBUILD Wordlist, alrededor de 700 palabras, es un ejemplo de este subconjunto. Estas reglas enfatizan la descripción del vocabulario como un hipertexto auto-contenido y altamente conectado [Leite 93]. Además, el uso de estos principios estimula fuertemente la reducción de la ambigüedad en el léxico. Es imposible examinar el significado de un término separado de su contexto [Parianou 01]. Por lo tanto, cuando la denotación de un término no requiere la ayuda de ningún contexto, es muy probable que dicho término pertenezca al vocabulario mínimo.

La Figura 4-3 provee un DER [Chen 76] del LEL, el cual muestra la estructura del léxico y muestra claramente la posible presencia de sinónimos y cómo se aplica el principio de circularidad.

4.3. El proceso de creación del LEL

4.3.1. Generalidades del proceso

El Léxico Extendido del Lenguaje se crea completando los casilleros en el Modelo del LEL (ver Figura 4-2) usando información obtenida del dominio de la aplicación. La intuición, apoyada con una buena comprensión del modelo del LEL, puede usarse para crear el léxico, pero esto puede o no llevar a un documento bien concebido. Con lo cual, se necesitan heurísticas para permitir que cualquiera pueda llevar a cabo este proceso con éxito y, además, para evitar debilidades que normalmente se presentan en LELs de aparentemente buena calidad. Estas debilidades van desde símbolos relevantes omitidos hasta la inserción innecesaria de algunos otros, así como la inclusión de exceso de detalles en las descripciones de símbolos o la falta de ellos. Las heurísticas presentadas en la sub-sección siguiente han sido bien probadas en el campo y diseñadas para reducir el impacto de toda fuente conocida de problemas en el LEL.

Se debe enfatizar que el LEL no es meramente una enumeración de entradas; su proceso de creación debe ser concebido como un enfoque comprensivo.

El proceso de creación del léxico, expuesto en la Figura 4-4 usando el modelo SADT[61] [Ross 77b], consiste de cinco actividades independientes: 1) *Planear*, 2) *Recolectar*, 3) *Describir*, 4) *Verificar* y 5) *Validar*, las cuales son descriptas en detalle en la siguiente sub-sección.

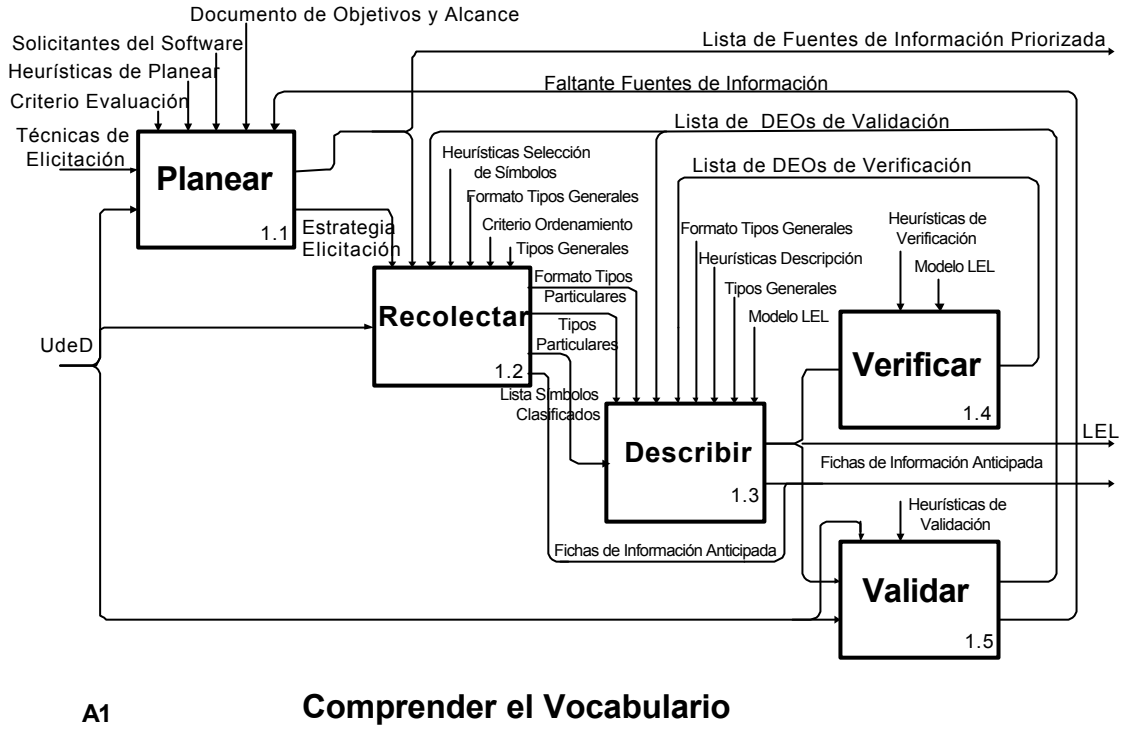


Figura 4-4. SADT del proceso de creación del LEL

Como se ve en la Figura 4-4, el proceso muestra un flujo principal compuesto de tres actividades: *Planear*, *Recolectar* y *Describir*, habiendo retroalimentaciones bien establecidas cuando la verificación y la validación tienen lugar. Después de verificar el LEL, el proceso retorna a la actividad *Describir*, donde se hacen correcciones basadas en la lista DEO[62] producida. Después de la actividad *Validar*, el proceso retorna a la actividad *Recolectar* y/o la actividad *Describir*, dependiendo de la lista DEO de validación producida, para hacer las correcciones necesarias. Para una fácil lectura, el modelo SADT no está mostrando todos los pasos de retroceso que pueden ocurrir durante el proceso de construcción. Por ejemplo, mientras se está describiendo un símbolo puede descubrirse que se le asignó mal el tipo, luego un paso atrás

[61] Notación del SADT: las cajas representan actividades; las flechas a la izquierda representan entradas requeridas por la actividad; las flechas hacia abajo representan controles; las flechas hacia arriba (no presentes en estos diagramas) representan mecanismos y las flechas a la derecha representan salidas desde la actividad.

[62] Una lista de DEO contiene las discrepancias, errores y omisiones encontradas durante las actividades de verificación y de validación, donde se sugieren correcciones.

ocurre para reclasificarlo (dentro de la actividad *Recolectar*). Otro ejemplo de retroceso puede deberse a la aparición de un nuevo término mientras se está describiendo otro. Es decir, la estrategia no es lineal, en vez, es un proceso iterativo donde la retroalimentación es un mecanismo constante. Además de esta retroalimentación continua, el flujo principal no sigue completamente un modelo de cascada dado que en la práctica real las tres actividades principales se superponen parcialmente. Por ejemplo, un símbolo puede describirse completamente mientras se identifican nuevas fuentes de información para clasificar o describir otros símbolos.

El léxico debe adaptarse para que refleje la última comprensión lograda del dominio de la aplicación; como tal evoluciona a medida que el proceso de la IR evoluciona. Las siguientes fases de SDRES también pueden producir una lista de DEO sugiriendo cambios y agregados en el LEL.

4.3.2. Actividades del proceso

A continuación se detallan las actividades graficadas en el SADT de la Figura 4-4, y para las actividades *Planear* y *Recolectar*, se incluye más detalle presentando un modelo de SADT para la descomposición de cada uno de ellas, ver las Figuras 4-5 y 4-6 respectivamente. En el caso de la actividad *Recolectar*, un nivel más bajo de detalle se muestra en la Figura 4-7 para la actividad *Clasificar*.

(1) Planear

Para establecer cómo elicitar información del UdeD, se deben seguir tres pasos: primero identificar las fuentes de información, segundo evaluarlas, y finalmente seleccionar las estrategias para elicitar los símbolos. La Figura 4-5 muestra el flujo de estas tres actividades usando un modelo SADT. Los productos de esta fase son una lista de fuentes de información y de las estrategias seleccionadas.

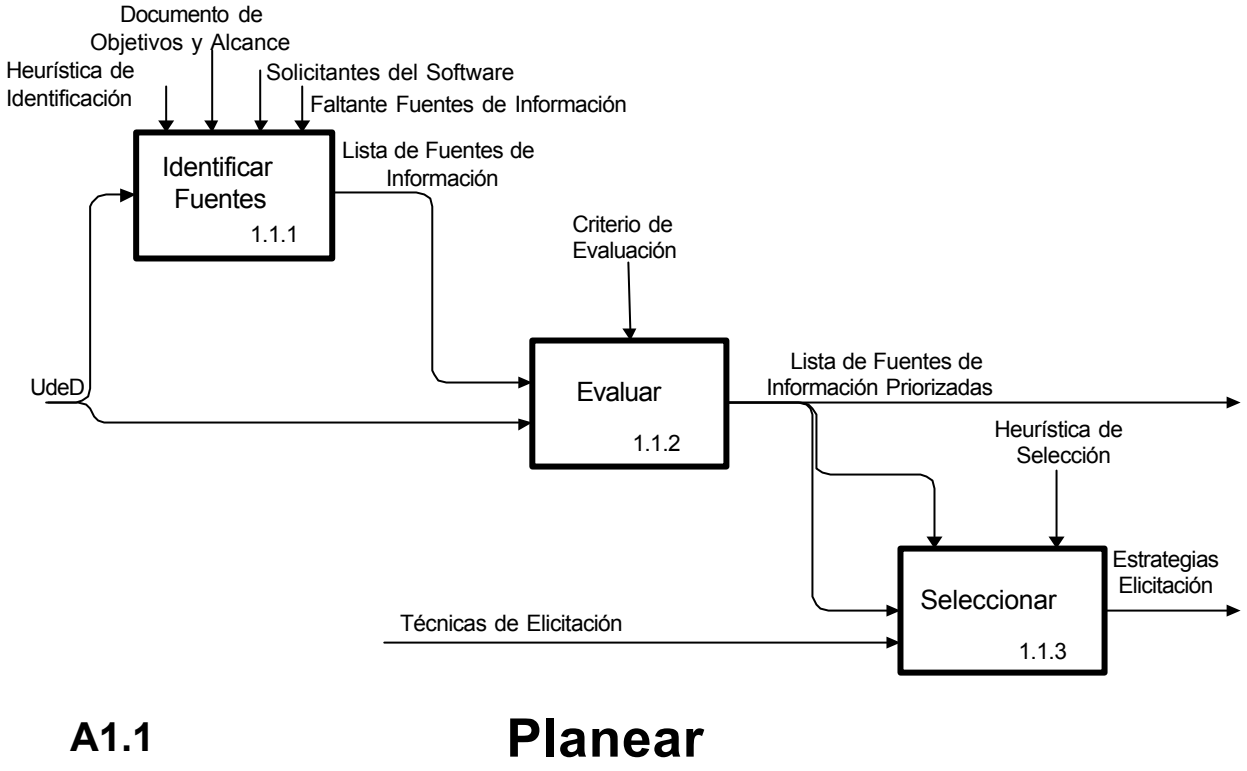


Figura 4-5 - SADT de la actividad Planear en el proceso del LEL

(1.1) Identificar Fuentes

Siguiendo SDRES, la creación del LEL es la primera actividad del proceso del software, entonces la identificación de las fuentes de información se vuelve un paso trascendente. En este paso se está considerando una visión amplia que abarca el proceso de desarrollo entero y, por lo tanto, se identifican las fuentes de información para todas las actividades de la IR y no sólo para la creación del LEL.

Como ya se mencionó en la sub-sección 2.2.4, las fuentes de información están contenidas en el UdeD, por lo tanto, el primer paso es definir el contexto dónde el proceso de IR tiene lugar, para ello se tiene como una fuente inicial el documento de objetivos y alcance del sistema (de existir) y los solicitantes del sistema del software. A partir de éstas comienza el descubrimiento de nuevas fuentes de información, armándose una lista con ellas. Al finalizar esta actividad deben poder responderse las siguientes preguntas:

- ✓ ¿ Quién es el cliente ?
- ✓ ¿ Quiénes serán los usuarios ?
- ✓ ¿ Quiénes son usuarios “claves” ?
- ✓ ¿ Existe alguna solución (paquete) ya disponible en el mercado ?

- ✓ ¿ Existe la posibilidad de reusar software ?
- ✓ ¿ Cuáles son los documentos más referenciados por los actores del UdeD ?
- ✓ ¿ Cuáles son los libros relacionados con la aplicación ?

(1.2) Evaluar

Después de identificar las fuentes de información, deben darse prioridades a las mismas, principalmente cuando se dispone de un gran número de fuentes dado que en algunos casos es casi impracticable el acceso a todas ellas. Leite [Leite 94] ha presentado heurísticas generales para identificar y dar prioridad a las fuentes de información, y en [Leite07] se presenta una estrategia detallada para identificar fuentes de información, priorizarlas y asignarles técnicas de elicitación. Básicamente, estas heurísticas proponen identificar primero a los solicitantes del software, luego a las personas que se presume serán impactadas por la operación del software y finalmente a cualquier documento relacionado con el UdeD. Existen varias propuestas a tal fin como se mencionó en la sub-sección 2.2.4.

Una fuente de información puede considerarse desde varios puntos de vista. En esta tesis, se distingue el punto de vista de la vigencia que clasifica la información en actual y formal. La información formal es acerca de lo que debe hacerse o debe ocurrir, pero no necesariamente información actualizada o acerca de la práctica corriente. Mientras que la información actual involucra prácticas o estados corrientes, es decir, lo que realmente está en uso.

Este tipo de punto de vista está muy cercano a la naturaleza de la fuente de información. Es decir, los usuarios, los formularios y el software actual son ciertamente fuentes que proporcionan información con un sesgo a lo actual, mientras que los manuales de procedimientos y las políticas del negocio presentan información con un sesgo formal. En este último caso, debe controlarse la información para identificar cuál es información actual ("lo que es"), información obsoleta ("lo que ya no es") o un tercer tipo: información no obsoleta pero en desuso ("lo que debe ser pero no es"). La información obsoleta no debe ser completamente desechada pues un efecto de la etimología puede existir en algunos símbolos que pueden facilitar la comprensión de su significado actual[63].

(1.3) Seleccionar

Este paso apunta a seleccionar la estrategia o conjunto de estrategias más adecuadas para elicitar los símbolos del dominio de la aplicación. Las técnicas principales para recolectar información, como ya se mencionó en la sub-sección 2.2.4, son lectura de documentos, entrevistas, observaciones, encuestas, reuniones, enfoque antropológico, reuso de requisitos, recupero

[63] Un nombre de símbolo proviene bastante frecuentemente de un contexto caduco pero que se usa ahora con nuevas denotaciones o connotaciones. La comprensión del significado actual del símbolo puede facilitarse teniendo en cuenta la etimología del nombre.

desde el diseño del software, entre otras. Pero para la creación del LEL, algunas de ellas son menos útiles. La etnografía ayuda a los ingenieros de requisitos a sumergirse en la cultura del UdeD facilitándole la comprensión del vocabulario pero agregándole el riesgo de dificultarle después el establecer qué palabras o frases son peculiares o muy repetidas. El recupero desde el diseño del software puede contener mucho vocabulario computacional. La observación es una técnica excelente para capturar el comportamiento actual pero no tan apropiada para capturar el vocabulario. El uso del resto de las estrategias depende principalmente de las fuentes de información previamente identificadas y de los objetivos y alcance del sistema, aunque las entrevistas y la lectura de documentos son los mecanismos más comunes para recolectar símbolos.

Las entrevistas permiten a los ingenieros reconocer fácilmente el vocabulario que los clientes y usuarios emplean en su ambiente. Se recomienda el uso de entrevistas no estructuradas (también llamadas entrevistas abiertas), haciendo preguntas sólo para motivar a que los clientes y usuarios hablen. La técnica entrevista normalmente se combina con la lectura de texto, principalmente formularios, comprobantes, informes, manuales y políticas de la organización.

(2) Recolectar

Para identificar y registrar los símbolos del lenguaje de la aplicación, deben seguirse tres pasos: primero identificar los símbolos, segundo organizarlos en una lista y finalmente clasificarlos. La Figura 4-6 muestra el flujo de estas tres actividades usando un modelo SADT. El producto de esta fase es una lista de símbolos.

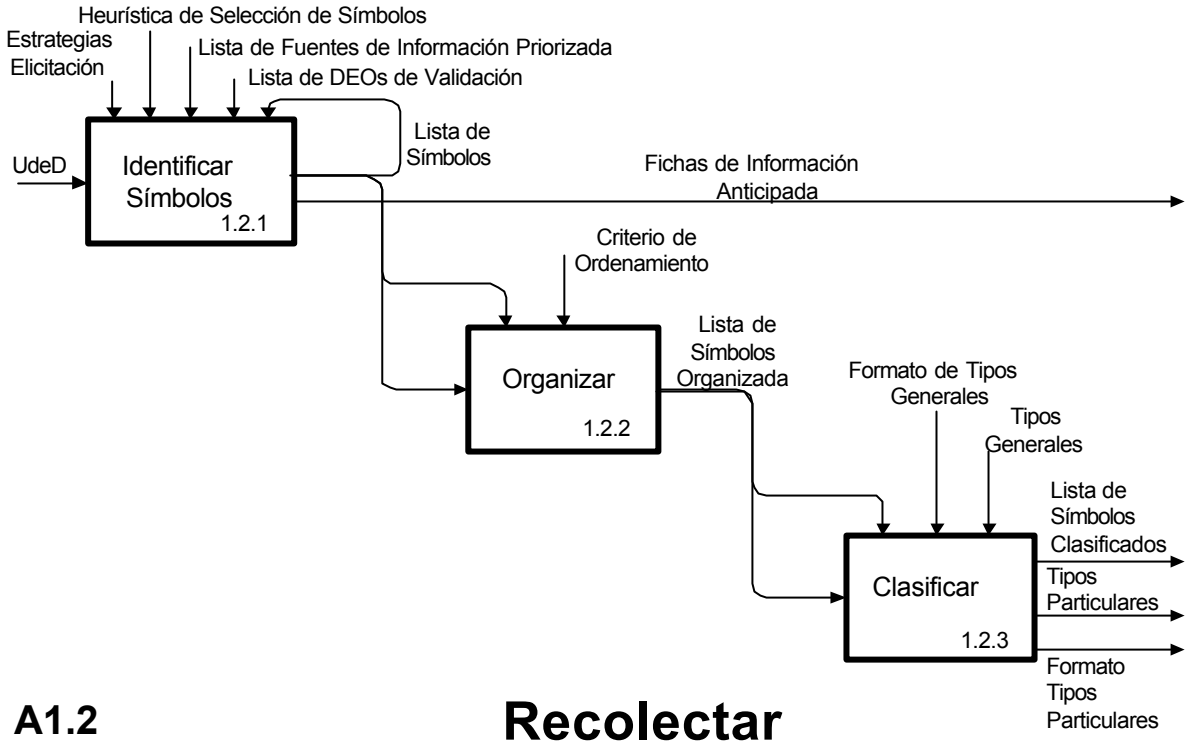


Figura 4-6. SADT de la actividad Recolectar en el proceso del LEL

(2.1) Identificar Símbolos

Esta tarea apunta a capturar los símbolos del léxico del UdeD aplicando las estrategias seleccionadas a las fuentes de información identificadas. Los ingenieros de requisitos identifican esos símbolos y construyen una lista de símbolos candidatos.

Como se ha mencionado, algunas técnicas para la captura de información, como la observación o el recupero desde el diseño del software, pueden no proporcionar la mejor información para recolectar símbolos. Este paquete de información puede examinarse con un criterio diferente en las restantes actividades de definición de requisitos. También, las entrevistas y la lectura de documentos pueden proporcionar más información que la actualmente necesaria.

Toda información extra que implique un pedido o necesidad referida al software deberá registrarse en las Fichas de Información Anticipada (ver sección 4.5).

Los documentos pueden estar disponibles al principio del proceso, sin embargo se obtienen habitualmente durante y después de las primeras entrevistas.

Normalmente, las primeras entrevistas son no estructuradas, dejando que los clientes y usuarios se expresen libremente. En las siguientes entrevistas, se recomienda generar una lista de preguntas guiadas para profundizar en los temas y ahorrar tiempo. El objetivo principal de las primeras entrevistas es recoger palabras o frases con un significado especial en el dominio de la aplicación, pero sin entender su semántica. Las entrevistas generalmente se llevan a cabo en el lugar de trabajo de los clientes y usuarios. El número de entrevistas depende principalmente de:

- ⇨ la complejidad del problema;
- ⇨ la experiencia de los ingenieros de requisitos creando un léxico;
- ⇨ el conocimiento de los clientes y usuarios sobre el UdeD;
- ⇨ el número de clientes y usuarios seleccionados para entrevistar.

Cuando se usan fuentes de información humanas, toda la información capturada debe registrarse (grabando, filmando o transcribiendo) teniendo en cuenta una posible indisponibilidad de esa fuente posteriormente.

Debajo se enumeran algunas reglas para seleccionar símbolos:

- ✓ Seleccione exclusivamente palabras o frases pertenecientes al dominio de la aplicación.
- ✓ Seleccione palabras o frases frecuentemente usadas por los clientes y usuarios o con alta repetición en los documentos.
- ✓ Seleccione palabras o frases “significativas” en el dominio de la aplicación.
- ✓ Excluya palabras o frases demasiado obvias que son de dominio público y no particulares del dominio de la aplicación.
- ✓ Considere aquellas palabras o frases que parecen estar fuera de contexto, desconocidas o confusas como términos dudosos para luego ahondar en ellos
- ✓ Identifique el nombre completo del término no importa cuán largo sea. Sin embargo, una abreviatura o un acrónimo puede ser el nombre de un término.
- ✓ Tenga presente que una abreviatura, un acrónimo o un nombre parcial pueden ser un sinónimo de un símbolo con un nombre largo.
- ✓ En caso de vincular una abreviatura, un acrónimo o un nombre parcial como sinónimo de un símbolo con un nombre largo, asegúrese de su existencia en el UdeD.
- ✓ Al emplear la técnica de lectura de documentos, escoja los símbolos dentro del alcance de un párrafo, en lugar del texto entero.

La recomendación principal para este paso es “evite excluir términos antes de tiempo”, pues recién se está comenzando con la actividad de elicitar información.

Aunque obvia, la siguiente recomendación es siempre oportuna: “tenga siempre presente el objetivo general del sistema”. Aún cuando éste fuese concebido impreciso al inicio, sirve para guiar el proceso de captura de

símbolos, concentrando la actividad en la adquisición de información útil para el estudio del UdeD con una visión orientada por dicho objetivo[64].

(2.2) Organizar

El objetivo de este paso es obtener una única lista de símbolos de las listas candidatas. Puede producirse más de una lista candidata por varias razones, como ser, listas parciales de diferentes ingenieros de requisitos con la misma fuente de información o listas parciales obtenidas de diferentes fuentes de información. Por consiguiente, esas varias listas deben unificarse en una.

Se debe recordar que todos los nombres dados a un término, es decir los sinónimos, son una sola entrada en el léxico. Por lo tanto, se debe organizar la lista de nombres del símbolo de manera tal que el nombre más frecuentemente usado en el dominio de la aplicación aparezca primero. Es decir, si uno de los nombres del símbolo raramente se usa, éste debe ubicarse al final.

Luego, la lista de símbolos se clasifica alfabéticamente para facilitar la búsqueda de símbolos.

(2.3) Clasificar

La tarea de clasificación ayuda a la integridad y homogeneidad de las descripciones de los símbolos. Los siguientes pasos deben realizarse: establecer una clasificación, definir cada tipo de la clasificación y aplicarlo a los símbolos, produciendo una lista de símbolos clasificada. Establecer una clasificación significa adoptar la clasificación estándar o crear una nueva ad-hoc. La segunda opción no es necesaria en la mayoría de los casos y sólo debe aplicarse en tales dominios específicos que hacen inutilizable la clasificación general. El modelo SADT de la Figura 4-7 describe en detalle la actividad *Clasificar*.

[64] La experiencia adquirida muestra que es indispensable reforzar en los integrantes noveles el tipo de información a elicitar. Un ejemplo muy común se presenta al recolectar información para un sistema de producción con el fin de controlar el estado de los materiales, productos y procesos, pero en la práctica se obtiene un cúmulo de información sobre las maquinarias y su funcionamiento interno que es de poco valor para la gestión de control, pudiendo ser útil dicho acopio si se apuntase a la automatización de procesos.

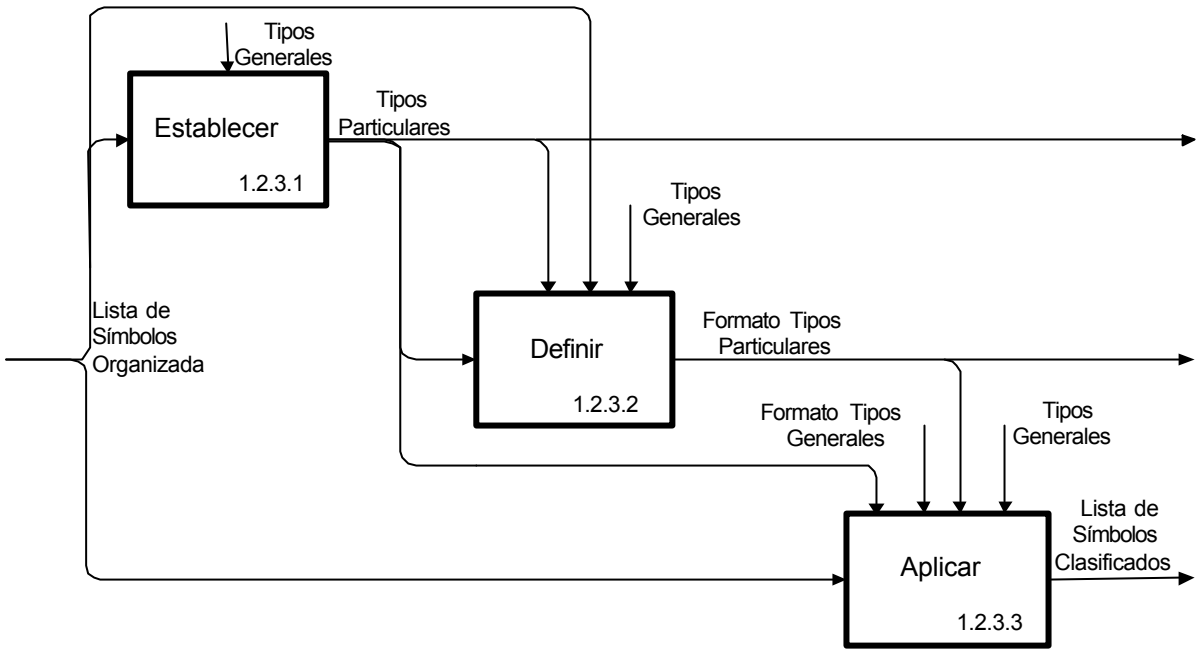


Figura 4-7. SADT de la actividad Clasificar en el proceso del LEL

(2.3.1) Establecer

El criterio de clasificación está dado por la clasificación general propuesta en [Leite 90] o por una clasificación particular. En la mayoría de los casos, la clasificación general se adecua perfectamente dando suficiente soporte para producir un léxico homogéneo con la información apropiada.

La clasificación general agrupa los símbolos en cuatro tipos: Sujeto, Verbo, Objeto y Estado. Estos tipos representan cada uno lo siguiente:

- ⇨ **Tipo Sujeto**: una entidad activa, por ejemplo, una persona, organización, máquina o sistema, que realiza actividades en el dominio de la aplicación.
- ⇨ **Tipo Objeto**: una entidad pasiva a la que se le aplican acciones en el dominio de la aplicación, sin realizar acciones por sí misma.
- ⇨ **Tipo Verbo**: una actividad o acción que ocurre en el dominio de la aplicación.
- ⇨ **Tipo Estado**: una condición o situación en la cual sujetos, objetos o verbos del dominio de la aplicación están o pueden estar en un momento dado.

Cuando se tiene un gran número de símbolos y/o un equipo de ingenieros de requisitos, puede considerarse refinar algunos tipos de la clasificación general o crear algunos nuevos, ajustando la clasificación al dominio específico de la aplicación. Esto ayudaría a hacer más precisa la definición de cada símbolo y por consiguiente, a garantizar una sintaxis homogénea del léxico. En este caso, deben definirse cuidadosamente los tipos particulares basándose en la lista de símbolos. Esta circunstancia puede tener lugar cuando parte del equipo de ingenieros tiene poca experiencia en la tarea, por lo tanto, requieren más ayuda, la cual puede proporcionarse introduciendo tipos basados en el dominio de la aplicación. Por ejemplo, los símbolos que comparten algunos atributos pueden describirse mejor usando una plantilla diseñada exclusivamente para ellos, en lugar de usar la plantilla del tipo general correspondiente.

(2.3.2) Definir

Las plantillas para definir los cuatro tipos generales [Leite 93b] son:

- ✓ Para un **símbolo Sujeto**, la noción tiene que definir quién es y el impacto debe registrar las responsabilidades o actividades que ejecuta o recibe.
- ✓ Para un **símbolo Objeto**, la noción debe definir qué representa, sus características e identificar otros objetos con los que se relaciona. El impacto debe describir las actividades o acciones que se aplican al objeto o que se realizan con él.
- ✓ Para un **símbolo Verbo**, la noción debe describir concisamente la actividad que representa, su propósito e identificar quién lo ejecuta, cuándo (cronológicamente o relativo a otras actividades) y dónde se realiza. El impacto debe registrar las acciones, operaciones o procedimientos involucrados en la actividad y debe identificar las situaciones que impiden la ocurrencia de la actividad, qué otras actividades se desencadenan en el UdeD y qué situaciones ocurren debido a esta actividad.
- ✓ Para un **símbolo Estado**, la noción tiene que hacer claro lo que significa y los estados y actividades o acciones que llevaron a este estado. El impacto debe identificar otros estados y actividades o acciones que pueden ocurrir a partir de este estado.

Si una clasificación particular ha sido escogida entonces se debe definir cada tipo particular. Esta definición consiste en describir con precisión la plantilla de cada nuevo tipo, indicando la clase de contenido tanto en la noción como en el impacto.

Las plantillas de los tipos generales pueden usarse como guía para definir los tipos particulares. Estos deben ser al menos tan detallados como los tipos generales.

(2.3.3) Aplicar

Una vez que la clasificación ha sido fijada o adaptada, debe aplicarse un tipo a cada símbolo en la lista. A menudo, esta tarea se realiza mientras se define la clasificación. Después de esto, la lista de símbolos que fuera previamente organizada, presenta un tipo para cada símbolo.

En la mayoría de los casos, hay muy pocos símbolos que pertenecen al tipo Estado, dado que usualmente hay un bajo nivel de abstracción dentro del dominio de la aplicación. Generalmente, los calificadores de sujetos, objetos o verbos reemplazan a los estados y, por lo tanto, no son usados por los clientes y usuarios. Los ingenieros de requisitos no deben crear estados si realmente no existen. Si el dominio de la aplicación ha creado por sí mismo la abstracción del estado, debe registrarse en el LEL, sino los ingenieros tienen que registrar los calificadores como parte del nombre de los sujetos, objetos o verbos. Cuando un calificador se encuentra en más de un símbolo del LEL, esto puede sugerir la existencia de un estado. El calificador podría ser un símbolo Estado por sí mismo. Por otro lado, el calificador más el sujeto / objeto / verbo puede genuinamente representar un nuevo tipo de sujeto / objeto / verbo. Se debe recordar que el LEL debe mostrar el uso del símbolo en el UdeD.

Debe prestarse atención especial también a los símbolos Verbo dado que pueden aparecer de diferentes maneras. A continuación, se detallan reglas para ajustar los nombres de símbolos de verbos (heurísticas de ajuste):

- ✓ Use el modo infinitivo para el nombre de los símbolos Verbo.
- ✓ Transforme las frases verbales de la voz pasiva a la voz activa.
- ✓ Cuando el nombre de un símbolo Verbo puede ser un verbo o su forma sustantiva, incluya ambos nombres como sinónimos siempre que ambos se usen realmente en el dominio de la aplicación.

Al tratar con un símbolo cuyo nombre es un sustantivo (independientemente de su tipo), use la forma singular siempre que sea posible. Esto no debe aplicarse en ciertos casos especiales:

- ✓ Las versiones singulares y plurales no se usan con el mismo significado.
- ✓ La forma singular no tiene sentido en el UdeD.

(3) Describir

Describir los símbolos implica la definición de su noción y su impacto, usando el Modelo del LEL y la plantilla del tipo al que pertenecen. Para describirlos, los ingenieros de requisitos pueden usar conocimiento previamente elicitado, aunque casi siempre deben retornar al UdeD para capturar más información. En este caso, es recomendable conducir entrevistas estructuradas para indagar a los usuarios sobre el significado de los símbolos, aunque pueden usarse otras fuentes de información, ver el paso ***(1)***.

A continuación se detallan reglas para describir los símbolos en el léxico (heurísticas de representación):

- ✓ Cada símbolo debe tener al menos un nombre, una oración en la noción y una oración en el impacto. Es decir, no se permite una entidad nula.
- ✓ Cada nombre del símbolo debe ser el usado en el dominio de la aplicación, a pesar de su longitud.
- ✓ Los símbolos usados como sinónimos en el dominio de la aplicación deben compartir una entrada en el LEL.
- ✓ Los símbolos que tienen también un significado común deben contener sólo el significado que se usa en el dominio de la aplicación.
- ✓ Elimine del vocabulario mínimo los símbolos redefinidos en el LEL. Así se usarán sólo con el significado del dominio de la aplicación.
- ✓ La noción y el impacto deben describirse usando oraciones simples y directas.
- ✓ Las oraciones deben expresar una sola idea.
- ✓ Las oraciones deben contener un solo verbo.
- ✓ Las oraciones de la noción y del impacto deben obedecer los principios de circularidad y de vocabulario mínimo.
- ✓ Cada oración de la noción o del impacto debe escribirse de forma tal que permita identificar la vigencia de la información involucrada: formal o actual, ver el paso ***(1.2)***. Se debe registrar ambos tipos de información[65].
- ✓ La descripción de la noción y del impacto de cada símbolo debe ajustarse a la plantilla del tipo de símbolo, ver el paso ***(2.3.2)***.
- ✓ Si dos símbolos comparten una misma oración en la noción o impacto, debe repetirse en ambas entradas.
- ✓ Toda noción e impacto debe hacer referencia por lo menos a otro símbolo. Preferentemente toda oración que describe un símbolo debería tener una referencia por lo menos a otro símbolo del LEL.
- ✓ Dado que un símbolo debe tener referencias a otros símbolos, dichas referencias deben marcarse (subrayando, resaltando o por cualquier otro medio).
- ✓ Todo símbolo debe ser referido por al menos otro símbolo.
- ✓ Cuando se hace referencia a otro símbolo, puede usarse cualquiera de sus nombres pero debe usarse el nombre completo.

Al agregar nuevos símbolos, por ejemplo debido a la retroalimentación proporcionada por los procesos de verificación o validación (ver los pasos ***(4)*** y ***(5)***), debe tenerse en cuenta cuidadosamente la siguiente regla:

- ✓ Se debe repasar el LEL entero para que el nuevo símbolo sea referenciado por al menos otro símbolo.

Con respecto a la completitud (ver sub-sección 4.3.3) deben considerarse las guías siguientes para obtener un LEL uniformemente detallado:

[65] Es conocido que en algunos contextos la adherencia a lo formal genera rechazos al incorporarse al software. Se debe tener conciencia de esto y evaluar con la organización la política a seguir.

- ✓ Tenga cuidado con el nivel de detalle utilizado en las descripciones de los símbolos.
- ✓ Registre todo impacto conocido.
- ✓ Describa los impactos acentuando el "qué" de las acciones, principalmente en los símbolos del tipo sujeto u objeto.
- ✓ Evite declaraciones sobre el "cómo" de las acciones en los impactos.
- ✓ Evite la generalización en las descripciones, porque se puede perder una referencia a otro símbolo, sobre todo a un símbolo Verbo. Aunque, por otro lado, use el mecanismo de abstracción para describir los impactos, de manera tal de evitar un léxico orientado a los procedimientos.

Estas reglas requieren un cierto grado de abstracción en la descripción de los símbolos sin perder información pero tampoco sobre-especificándolos. Existe una tendencia a recargar de información el componente impacto por el afán de volcar en el modelo actual la mayor cantidad de información elicitada. Este exceso de información puede deberse a una elicitación, a veces, no encauzada correctamente al objetivo de la etapa, donde el ingeniero de requisitos se aboca a entender el problema y sus detalles, o cuando se cuenta con usuarios que brindan más información de la solicitada, ya sea por su personalidad o por desarrollar una actividad en niveles operativos bajos. Cuando esta información no ayuda a comprender mejor un símbolo, significa que ella debe registrarse en la Ficha de Información Anticipada. Cabe recordar que en estas fichas también debe volcarse información referida a problemas o necesidades de los usuarios. En síntesis, toda información referida a "cómo" se desarrollan las actividades actualmente o relativa a los problemas existentes y las necesidades a satisfacer en el futuro debe quedar asentada en dichas fichas.

Por otro lado, en este paso, los símbolos deben explorarse para descubrir más símbolos o unificar sinónimos no detectados previamente como tales. La principal fuente en esta exploración son los símbolos cuyo nombre se construye con una frase. En este caso, se debe identificar el sustantivo o verbo que constituye el núcleo del símbolo. La lista entera de símbolos debe explorarse buscando:

- ✓ Símbolos con el mismo núcleo y diferente modificador.
- ✓ Símbolos con un núcleo solamente.

Si hay un grupo de símbolos que comparten el mismo núcleo, se debe considerar la existencia de una relación jerárquica entre ellos (ver sub-sección 4.3.3):

- ✓ Si no hay ningún símbolo con el núcleo solo, examine la existencia del núcleo en el vocabulario del UdeD.
- ✓ Si se encontró un símbolo con el núcleo solo, considere las siguientes variantes:
  - o El símbolo con el núcleo solo es una generalización de algunos de los otros símbolos.

  - El símbolo con el núcleo solo se usa como sinónimo de algunos de los otros símbolos.
- ✓ Examine la existencia de símbolos no descubiertos en el dominio de la aplicación con el mismo núcleo más otros modificadores.
- ✓ Asegúrese que no haya sinónimos entre estos símbolos.

Las reglas precedentes ayudan a descubrir omisiones de símbolos y a capturar sinónimos, pero no deben inducir a la creación de términos.

(4) Verificar

La actividad de verificación del LEL se realiza siguiendo heurísticas que facilitan la detección de defectos. Se obtiene a partir de esta actividad una lista de DEOs. Los aspectos principales que cubre la verificación son:

⇨ Verificar la sintaxis:
- ✓ detectar el uso de sintaxis no uniforme entre varios símbolos.
- ✓ controlar el nombre de los símbolos en singular.
- ✓ controlar el nombre de símbolos del tipo Verbo en modo infinitivo o en su forma sustantiva.

⇨ Verificar los componentes:
- ✓ controlar la falta de noción o impacto en los símbolos.
- ✓ controlar sinónimos ocultos: dos o más símbolos con descripciones solapadas.

⇨ Verificar la clasificación de símbolos:
- ✓ controlar el tipo asignado a cada símbolo.
- ✓ comparar la noción con la plantilla del tipo de símbolo asignado.
- ✓ comparar el impacto con la plantilla del tipo de símbolo asignado.

⇨ Verificar el uso de los símbolos del léxico:
- ✓ controlar el marcado de todas las referencias a símbolos en cada noción e impacto.
- ✓ controlar el uso del nombre completo del símbolo cuando se hace referencia a él.
- ✓ buscar oraciones sin ninguna referencia a símbolos.
- ✓ controlar la semántica del uso de símbolos al describir otros símbolos.

⇨ Verificar la adherencia a los principios:
- ✓ cumplir el principio de circularidad en la descripción de cada símbolo.
- ✓ cumplir el adherencia al principio de vocabulario mínimo en la descripción de cada símbolo.

⇨ Verificar las relaciones jerárquicas (ver sub-sección 4.3.3):

- ✓ controlar la inclusión de la noción de un símbolo en otros símbolos.
- ✓ controlar la inclusión del impacto de un símbolo en otros símbolos.
- ✓ controlar la coherencia en las descripciones de un símbolo "super-tipo" respecto a sus símbolos "sub-tipo".

Después de verificar el LEL, la lista de DEOs retroalimenta la actividad Describir donde los autores del léxico producen las correcciones necesarias. En esta tesis se recomienda como técnica de verificación del LEL: la inspección, ver sección 6.2 donde se la detalla aplicándola a escenarios y en [Kaplan 00] se la aplica al LEL.

(5) Validar

Mientras se identifican y describen símbolos tiene lugar una validación informal. Posteriormente, la actividad formal de Validar permite que los ingenieros corrijan, ratifiquen o mejoren el conocimiento sobre el vocabulario del dominio de la aplicación. Las validaciones informales apuntan a controlar la lista candidata. Validaciones posteriores se usan para controlar el conocimiento adquirido y para mejorar el léxico clarificando dudas, corrigiendo las definiciones de símbolos y excluyendo o agregando símbolos.

La tarea de validación normalmente consiste en entrevistas estructuradas o reuniones con los usuarios en su lugar de trabajo. No todos los símbolos se controlan contra cada usuario, sólo se validan con él aquellos símbolos relacionados con el contexto del entrevistado. Se puede leer la descripción de cada símbolo a los entrevistados, quienes confirman, corrija, hacen observaciones o agregan información faltante. A veces, en lugar de leer las descripciones del símbolo durante la entrevista, el ingeniero entrega una copia del LEL entero o la parte de interés para el entrevistado antes de la entrevista. La validación se lleva a cabo fácilmente con los usuarios porque no tienen dificultad en la comprensión del léxico dado que está escrito en LN y empleando su propio vocabulario.

Resumiendo, el proceso de validación apunta básicamente a:

- ⇨ corregir nociones e impactos de símbolos ya descriptos;
- ⇨ ratificar la definición de los símbolos;
- ⇨ identificar nuevos símbolos, sinónimos y homónimos.

El proceso de validación genera una lista de DEOs similar a la producida en la verificación. La lista retroalimenta luego las actividades Recolectar y/o Describir, para hacer las correcciones necesarias. Si es necesario, este retroceso puede requerir elicitar nuevas fuentes de información en la actividad Planear.

4.3.3 Otros tópicos en el proceso

El proceso de creación del LEL parece ser un trabajo fácil donde cada actividad tiene un camino preciso y claro a seguir. Aunque esto es básicamente verdad, se puede presentar algún caso donde se dificulta decidir sobre ciertos aspectos. Ellos están principalmente relacionados con la completitud, la presencia de homónimos y los sinónimos parciales. Estos puntos aparecen como preguntas tales como: ¿Cuándo terminamos de describir los símbolos? ¿Cuándo dejamos de seleccionar símbolos? ¿Cómo tratamos a un símbolo que tiene dos significados diferentes dentro del UdeD? ¿Qué es lo mejor a hacer cuando varios términos tienen información solapada?

Un tratamiento descuidado de estas dudas conducirá a producir un LEL de baja calidad donde diferentes criterios se usarán frente a la ocurrencia del mismo problema. Para evitar esta variación de criterios dentro del LEL, se debe seguir una serie de pautas cada vez que acaece alguna de estas preguntas. En esta sub-sección, se presentan recomendaciones y, en algunos casos, se dejan problemas abiertos donde los ingenieros de requisitos deben tomar una decisión sobre el mejor tratamiento según el UdeD.

El problema de completitud.

La completitud en cualquier modelo es una cualidad deseable que rara vez es alcanzable y menos aún validable. Por ello, se han propuesto para muchos modelos reglas para determinar cuándo se considera que el modelo es lo más completo posible [Pitts 04]. Frecuentemente, ese logro depende del objetivo asumido para dicho modelo.

Se pueden distinguir dos tipos de completitud en el LEL: interna y externa [Doorn 03]. La primera se refiere a la cantidad de información utilizada para describir un símbolo, y la segunda a la cantidad de símbolos que conforman el LEL.

La completitud externa concierne entonces con garantizar que el LEL tiene todos los símbolos necesarios y suficientes para comprender el vocabulario del UdeD. Pero como ya se ha mencionado, esto es casi imposible de asegurar en cualquier modelo. Lo que más atenta contra ello es la dificultad de determinar cuáles son términos del LEL y cuáles no, bajo el supuesto de cuáles son relevantes o no para el UdeD, siendo que éste aún no se conoce. La dificultad se agrava si se agrega otra condición: poder establecer para cuáles es valioso disponer o no de una definición asumiendo un conocimiento tácito que puede llevar a malas interpretaciones. Con las heurísticas de selección y de descripción de símbolos (ver sub-sección 4.3.2: actividades *Recolectar* y *Describir*), se considera que este dilema puede reducirse notoriamente.

Doorn y Ridao [Doorn 03] aplicaron el método Detection Profile Method [Wohlin 98] para estimar la cantidad máxima de símbolos relevantes en un UdeD. Para ello utilizaron la cantidad de símbolos identificados en LELs creados por varios grupos independientes de ingenieros de requisitos que

habían aplicado la misma técnica de elicitación sobre un mismo UdeD. Lograron confirmar la hipótesis de [Wohlin 98], pues al aumentar la cantidad de grupos elicitadores la brecha entre la cantidad estimada y la cantidad detectada de símbolos decrece.

La completitud interna en el LEL está vinculada al contenido descriptivo de los símbolos en la noción e impacto, luego se debe poder asegurar que cada símbolo contiene la información necesaria y suficiente para su comprensión. Si se considera al LEL como un diccionario pequeño, también se lo debe reconocer como un diccionario de uso [Fowler 83] [Garner 98]. Las raíces de tal paralelismo descansan en el hecho que el impacto en el LEL es similar a la porción de uso de estos diccionarios. Entonces, ¿el impacto se incluye para clarificar y dar contexto a la noción o, es un componente con problemas propios de completitud?

Para contestar adecuadamente a esta pregunta, se debe repasar el propósito del LEL. La idea es entender el vocabulario del UdeD. Por lo tanto, ¿la falta de un impacto es una pérdida y esta omisión hace más débil al LEL? No hay una respuesta claramente afirmativa a estas preguntas. El impacto de un símbolo debe ser suficiente para definir el "perfil del símbolo". Hay una diferencia sutil entre una definición clara del perfil del símbolo y la búsqueda de completitud. La completitud no es un objetivo para el impacto pero se debe cuidar de no omitir impactos relevantes. Como ha observado Loucopoulos [Loucopoulos 95], el aspecto de completitud se refiere a no omitir información esencial mientras que el aspecto de minimalidad se refiere a incluir la información necesaria, esto es, evitar la sobre-especificación.

El contexto en cual el LEL se está creando agrega una dificultad extra a esto, sobre todo cuando el LEL se usa luego en el proceso de IR como una posible fuente de información para ayudar a la construcción de otros modelos. Siguiendo SDRES, los ingenieros de requisitos que están creando un LEL y saben que van a usarlo para derivar escenarios, están más inclinados a introducir un sesgo procedural en los impactos, pero esto debe evitarse. Otros autores [Sábat Neto 00] [Cysneiros 04] llevan su propuesta un paso más adelante, introduciendo en el LEL información adicional con un punto de vista predefinido y teniendo la completitud como un objetivo; esto lleva a léxicos especializados como el NFR-LEL de Cysneiros y Yu.

Jerarquías en el LEL.

El análisis de léxicos disponibles muestra la existencia de jerarquías de términos. Estas jerarquías están principalmente compuestas por símbolos del tipo sujeto y objeto, y en una menor proporción del tipo verbo, aunque es perfectamente posible una jerarquía para símbolos del tipo estado. Es decir, se puede observa una relación "tipo / super-tipo" entre los símbolos. En el campo lingüístico, esta relación se denomina "hiperonimia" ("es-un") y la contrapartida es "hiponimia" ("ejemplar-de") [Ureña 01]. Algunos investigadores en la IR han comentado este tipo de relación entre actores del UdeD. Por ejemplo, Constantine [Constantine 98] ha denominado "especialización" a esta relación

mientras que Schneider [Schneider 98] la llama "herencia entre actores". La Figura 4-8 muestra cuatro ejemplos de jerarquías para símbolos de los tipos sujeto, objeto y verbo.

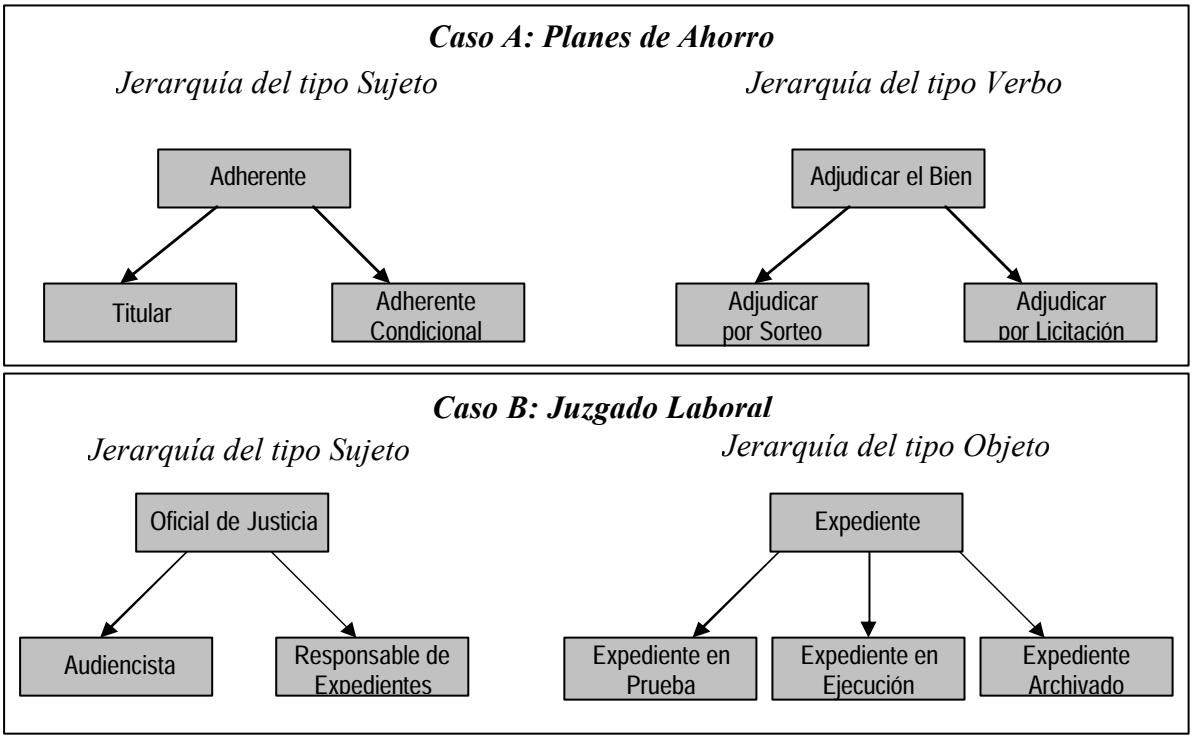


Figura 4-8. Ejemplos de jerarquías de símbolos[66]

La detección de esta relación jerárquica debe ayudar a reducir los significados ambiguos de los términos en un contexto dado y a completar o incrementar el conocimiento sobre aquellos términos con menor abstracción respecto de aquéllos en los niveles altos de la jerarquía.

En muchos léxicos se encuentran referencias (vínculos internos) a términos inexistentes, aún en léxicos creados por ingenieros experimentados. Esto frecuentemente ocurre debido a relaciones jerárquicas no detectadas; ver sub-sección 4.3.2 paso ***(3)*** *Describir.* Por ejemplo, en un LEL que contenía los símbolos "Título de Magíster" y "Título de Doctor" se encontró más de una referencia a la palabra "Título" [67] que no correspondía a ningún símbolo.

Para cada símbolo en el léxico, siempre existe un término más amplio, llamado género, que puede o no pertenecer al vocabulario del dominio de la aplicación. Las jerarquías entre los símbolos de LEL aparecen cuando dos símbolos comparten el mismo género y éste además es un símbolo del LEL.

[66] Los ejemplos del Caso A se extrajeron del LEL del proyecto "Sistema de Planes de Ahorro" desarrollado por el grupo INTIA de la UNCPBA en 1997. Los ejemplos del Caso B pertenecen al LEL del proyecto "Juzgado Laboral" basado en un caso real desarrollado por alumnos de 4º año de la carrera de Ingeniería en Sistemas de la UTN en el 2003 durante el curso de grado Ingeniería de Requisitos.

[67] Este ejemplo se extrajo del LEL del Sistema de Post Graduados [Breitman 98] modelado y desarrollado para el Departamento de Informática de PUC-Río, 1997, http://www.inf.puc-rio.br/~dilbert.

Incluso cuando el género de uno o más símbolos no pertenece al léxico, debe ser tenido en cuenta en la descripción de la noción de los símbolos que lo contienen. Kovitz [Kovitz 98] claramente recomienda mencionar el género al escribir cualquier definición. En concordancia con Kovitz, se sugiere la siguiente regla de descripción:

- ✓ Incluya el género en cada noción.
- ✓ Si el género es un símbolo del LEL, analice las descripciones de todos los símbolos que componen la jerarquía para determinar la consistencia en dichas descripciones, respecto del símbolo género.

Símbolo	Noción del Símbolo	Género	¿Género es Símbolo?
Expediente	Es el documento originado por la presentación de una demanda.	documento	No
Expediente en Ejecución	Es un expediente sobre el que se ha dictado sentencia.	expediente	Sí
Oficial de Justicia	Es un empleado que trabaja en el juzgado.	empleado	No
Responsable de Expedientes	Es un oficial de justicia que tiene asignado expedientes.	oficial de justicia	Sí
Pasaporte	Es un documento emitido por la Policía Federal para salir del país.	documento	No
Solicitante	Es una persona que inicia un trámite relativo al pasaporte.	persona	No

Tabla 4-1. Ejemplos del género en símbolos

En la Tabla 4-1 se muestran ejemplos de símbolos del LEL con géneros que pertenecen o no al LEL. Los cuatro primeros símbolos pertenecen a la jerarquía del Caso B de la Figura 4-8 y los dos últimos corresponden a símbolos mostrados en la Figura 4-1. En la columna "Noción del Símbolo" se ha colocado una sola descripción, aquella que presenta el género.

Como se muestra en los ejemplos de la Figuras 4-9, no toda jerarquía sigue el patrón: núcleo + modificador, mencionado en el paso ***(3)*** *Describir* de la sub-sección 4.3.2. Es en general a través del género que pueden detectarse las jerarquías.

El análisis de relaciones de sinonimia e hiperonimia ayuda a reducir la ambigüedad del LN. Una vez creado el léxico, la búsqueda de jerarquías de símbolos puede ayudar a mejorar la descripción de símbolos mediante la comparación de nociones e impactos de aquellos símbolos pertenecientes a la misma jerarquía y, también, puede ayudar en la búsqueda de posibles términos faltantes.

Pueden aparecer jerarquías más complejas en un glosario pero ello no es tan frecuente. Aparecen por ejemplo cuando un término con "un núcleo más un modificador" también se especializa mediante un par de nuevos modificadores, creando dos o más entradas adicionales en el glosario.

Un estudio exhaustivo de éstas y otras relaciones léxicas y semánticas debe realizarse dentro del alcance de un glosario para mejorar la calidad del producto y la comprensión del UdeD.

Tratamiento de homónimos.

Los homónimos raramente aparecen dentro del mismo contexto particular pero si esto ocurre debe aplicarse un tratamiento especial. Los homónimos no deben desecharse, pues la regla principal de la técnica del LEL es preservar el vocabulario del dominio de la aplicación. A veces los ingenieros de requisitos se tientan a introducir nuevos nombres para evitar su propia confusión. Sin embargo, los usuarios seguirán utilizando estos nombres obligando a los ingenieros a reconocer sus diferentes significados.

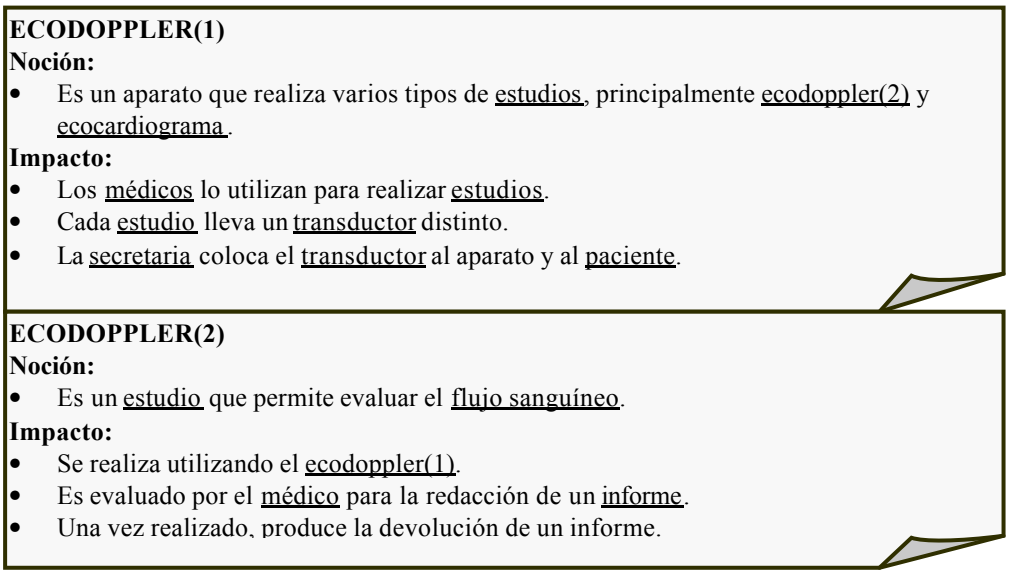


Figura 4-9. Ejemplo de homónimos[68]

Cuando el mismo término en el dominio de la aplicación se usa con dos o más significados, el término debe ser tratado como entradas diferentes en el léxico, dado que cada entrada muestra una definición del símbolo de acuerdo a un único significado dado por la noción y el impacto. Obviamente los nombres de entradas deben ser únicos como claves, por lo tanto, una manera de resolver los homónimos consiste en indexar el nombre repetido para cada entrada que representa un significado diferente. Entonces, cada referencia a un homónimo en otros símbolos debe distinguirse según su significado haciendo uso del índice en el nombre. Si se adopta una solución de indexado, las

[68] Este ejemplo fue extraído del LEL correspondiente al proyecto "Servicio de Ecodoppler y Ecocardiografía" de un sanatorio, desarrollado por alumnos de 4º año de la carrera Ingeniería en Sistemas de la UTN-FRBA en el 2005.

referencias al nombre deben ser siempre seguidas por el índice. La Figura 4-9 muestra un ejemplo de dos símbolos con igual nombre donde cada uno tiene un significado diferente.

ASIGNAR TURNO / ATENDER AL PACIENTE(1)
Noción:
- Es la acción de darle al paciente un día y horario para la realización de un estudio cardíaco.
- Es realizado por la secretaria.
- Es necesario para atender al paciente(2).

Impacto:
- Se registra en la agenda los datos del paciente y el horario asignado.
- El paciente puede solicitar el turno en forma personal o telefónica.

ATENDER AL PACIENTE(2)
Noción:
- Es la acción que consiste en realizarle un estudio al paciente.
- Es realizado por el médico o el técnico.
- Se realiza en un día y horario acordado previamente al asignar turno.

Impacto:
- El paciente debe presentar una orden de autorización.
- El médico o técnico realiza el estudio en el consultorio.
- El médico o técnico redacta un informe.

DEVOLVER INFORME / ATENDER AL PACIENTE(3)
Noción:
- Es la acción de entregar al paciente el resultado de un estudio.
- Lo realiza la secretaria por la mañana únicamente.
- Se realiza una vez que se atendió al paciente(2).

Impacto:
- Se le entrega al paciente el informe elaborado por el médico o el técnico.
- Se registra la entrega del informe.

Figura 4-10. Ejemplo de homónimos con sinónimos[69]

Es frecuente que la aparición de homónimos ocurra en distintas áreas de la organización, donde en un sector se emplea el término con un cierto significado diferente al de otro sector. Se aconseja identificar en la noción del símbolo el área o ámbito en el que dicho significado es empleado. Esto apunta a facilitar cada uno de los homónimos adecuadamente y dirigido a quienes así lo utilizan, y, por otro lado, ayuda a determinar con qué usuarios validar cada uno de estos homónimos.

Cabe observar que también es frecuente que estos símbolos presenten sinónimos que los distingan (ver Figura 4-10). En tal caso es aconsejable el uso de los nombres distintivos para una mayor claridad en la lectura de la documentación o modelos que hagan referencia a ellos. En realidad esta facilidad está dirigida al grupo de desarrollo pues se debe asumir que los clientes y usuarios son los propietarios de estos términos y por ende los

[69] Corresponde al mismo caso de la Figura 4-9.

entienden, los distinguen y saben utilizarlos apropiadamente.

Evolución del LEL.

El LEL no es el producto de una fase del desarrollo de software que se mantiene inmutable, sino por el contrario está en constante evolución. Mientras el proyecto avanza, crece el conocimiento acerca del UdeD y asimismo el LEL debe actualizarse. Pero además, el LEL evoluciona como consecuencia de la evolución del propio UdeD (ver sub-sección 1.1.4). Estos tipos de cambios en el LEL provocan la generación de versiones diferentes en la misma fase de desarrollo. La gestión de requisitos debe encargarse de esta evolución[70] (ver sub-sección 2.2.7).

Desde otro punto de vista, el LEL evoluciona dependiendo de la fase de desarrollo de software en curso conservando el propósito de su creación pero cambiando su contenido.

- ⇨ El LEL en la fase de requisitos refleja el vocabulario del dominio de la aplicación actual.
- ⇨ El LEL en la fase de diseño involucra nuevos términos y/o nuevas definiciones basadas en la repercusión del proyecto de software en el UdeD.
- ⇨ El LEL en el estado operativo refleja el vocabulario del nuevo dominio de la aplicación modificado por la implantación del software.

En los últimos dos ítems, la incorporación de nuevos símbolos en el LEL se debe principalmente a la aparición de nuevos recursos, nuevas operaciones o funciones y nuevos roles en el UdeD. Estos términos pudieron haber existido en el UdeD pero ahora toman relevancia en el UdeD modificado o pueden ser términos adoptados de otro contexto. Sin embargo, a veces el mecanismo de creación del término se torna importante. Es decir, los términos pueden surgir por diferentes mecanismos [Parianou 01b], tales como:

- ⇨ La adquisición de nuevo significado en términos existentes a través de la metáfora o metonimia;
- ⇨ La creación de nuevos términos por medio del préstamo desde otros idiomas, palabras compuestas o derivadas;
- ⇨ La creación de fraseologismos debido a su estabilidad sintáctica y semántica y su uso constante y común.

El conocimiento sobre los mecanismos de creación puede guiar para describir los términos en el léxico. Por ejemplo, cuando un término es un fraseologismo, el significado de la expresión no depende del significado de sus elementos constitutivos y por consiguiente debe evitarse el uso parcial del fraseologismo al describir otros términos.

[70] Aunque la actividad de la IR se denomina Gestión de Requisitos, ésta es mucho más amplia y abarca todos los cambios en los modelos construidos durante la IR, contengan o no requisitos. El LEL puede contener algunos requisitos del software aunque no es un modelo con tal fin.

Breitman & Leite [Breitman 98] estudiaron la evolución del LEL, mostrando tres versiones del léxico correspondientes a las fases de especificación, de diseño y de codificación respectivamente. La primera versión representa el vocabulario del dominio de la aplicación, la segunda incluye los cambios debido a las decisiones de diseño teniendo en cuenta como sería el dominio de la aplicación, mientras que la tercera versión del léxico contiene, por ejemplo, los nombres de objetos e interacciones entre ellos. Este uso del LEL se discute en la sección siguiente.

4.4. Observaciones

4.4.1. La técnica del LEL frente a la práctica de glosarios

En SDRES se considera la creación del LEL una actividad básica en la IR, para establecer una buena comunicación entre los involucrados y para anclar los términos utilizados en la descripción de los restantes artefactos construidos durante todo el proceso de desarrollo de software, mientras que en otras estrategias la construcción de un glosario es una actividad opcional [Constantine 98] [Jacobson 99].

SDRES enfatiza la creación de un glosario de términos del UdeD, a diferencia de otras propuestas que sólo proponen crear glosarios con términos empleados en documentos o modelos de requisitos para clarificar dichos términos exclusivamente, como por ejemplo en [Rolland 98c] [Alspaugh 99] [IEEE Std 830-1998].

Además, en esta tesis se han dado heurísticas de creación del glosario que incluyen un proceso de verificación mediante inspecciones y un proceso de validación. Otras propuestas mencionan meramente validaciones [Oberg 98] [Robertson 01] generalmente informales. Con la creación de un LEL bien verificado y validado, se pueden lograr descripciones con menos ambigüedades y más consistentes en todo documento y modelo producido, independientemente de las guías de estilo y contenido que acompañen la construcción de éstos últimos.

Respecto a la propuesta en [Alspaugh 99], esta tesis difiere de dicho enfoque principalmente porque en SDRES se recomienda la creación de un glosario unificado, antes de la construcción de los escenarios, y no un glosario por cada documento o modelo generado. Esto tiene más relación con la propuesta de Breitman & Leite [Breitman 98] de un glosario por cada fase del desarrollo (ver discusión en la siguiente sub-sección). Además, los términos del LEL se usan para describir los escenarios en un estilo consistente y orientado al usuario.

Varias estrategias que producen escenarios o casos de uso escritos en LN comienzan construyendo éstos y luego generan un glosario con los términos empleados en ellos para ayudar a su comprensión [Alspaugh 99] [Regnell 99b]. En esta tesis, se considera que es más fácil y natural comenzar

el proceso de requisitos comprendiendo el vocabulario del dominio del problema para luego comprender el problema mismo.

A diferencia de [Whitenack 94] [Robertson 01], la propuesta aquí es preservar el vocabulario tal cual se utiliza en el UdeD y esto incluye por lo tanto mantener todos los sinónimos y manejar los homónimos sin perder sus diferentes significados. Mientras que en [Kovitz 98] se propone distinguir los homónimos dentro de la misma entrada, la propuesta aquí es considerar cada significado como una entrada distinta, con el fin de distinguir dentro de cada documento o modelo producido el exacto significado al que se hace referencia.

4.4.2. Uso del LEL en otras estrategias

Desde su introducción [Leite 89], el uso del Léxico Extendido del Lenguaje se ha difundido en diferentes enfoques dentro de la IR y campos relacionados. Distintos investigadores adoptaron el LEL en su estructura básica mientras otros han modificado su estructura, su contenido e incluso adaptado la estrategia de creación para alcanzar otros propósitos.

Sábat Neto [Sábat Neto 00] creó un glosario, denominado NFR-LEL, que registra la terminología relativa a RNF. El contenido de este LEL define RNF primarios, RNF específicos y estrategias de satisfacción. Al crear un léxico de un dominio de aplicación particular, el NFR-LEL podría usarse como una lista de control para ayudar a la elicitación y modelado de más información de RNF en el LEL.

Como se mencionó en la sección 4.1, Breitman & Leite [Breitman 03] proponen un proceso de construcción de una ontología de aplicación web, que comienza creando un LEL de esa aplicación y luego construye la ontología extrayendo básicamente información del LEL. Debido a las propiedades del LEL (estructura, clasificación de símbolos e intra-vínculos), se facilita la tarea de identificar nuevos conceptos.

Leonardi [Leonardi 01] [Leite 98] muestra una estrategia para identificar reglas del negocio. Estas reglas se describen de acuerdo a patrones y se vinculan descripciones a las entradas del LEL previamente creado. El LEL no sólo se lo utiliza para explicar la terminología del negocio sino también para definir la estabilidad de las reglas.

Pimenta & Fausto [Pimenta 97] presentan un enfoque semiótico para elicitar requisitos de sistemas interactivos. La primera actividad de esta estrategia es la elicitación y la organización del conocimiento del lenguaje de los usuarios usando el modelo del LEL. Luego las descripciones de los símbolos del LEL se traducen a un modelo de objetos del dominio para detectar conceptos erróneos y omisiones en las definiciones y establecer las bases para la reusabilidad.

Fiorini et al. [Fiorini 00] presentan un enfoque sistemático para la descripción, almacenamiento y reuso de procesos, definiendo un marco de

trabajo del proceso y tipos de procesos. En particular, cuando describen patrones de proceso se marcan palabras y frases especiales y las vinculan a un LEL, el cual contiene definiciones del dominio. Usan el LEL para tener una mejor comprensión del dominio y para dar más semántica al lenguaje de descripción de procesos.

Ravid & Berry en [Ravid 00] proponen adaptar la interfaz de usuario y el proceso de construcción de prototipos de requisitos mediante la identificación y documentación de los tipos de información de requisitos que el prototipo contendrá, pues esto es un problema mayor en muchos proyectos donde el equipo de desarrollo supone que el prototipo contiene todo detalle del sistema, principalmente si ellos no tienen acceso a los ingenieros de requisitos. El prototipo se construye por medio de escenarios (que denominan “escenarios de casos de uso”). Simultáneamente se crea un LEL y se refleja en cada modelo construido, incluso en el propio prototipo, dado que el léxico los ayuda a la legibilidad de los modelos y a reducir ambigüedades. Además, los autores declaman que el LEL contiene información que no está incluida ni en los escenarios ni en el prototipo.

De Bortoli & Alencar Price [De Bortoli 00] proponen el modelado de flujos de trabajo para apoyar la elicitación de los requisitos. Durante la fase de elicitación se crea un LEL que describe el lenguaje del sistema de información, con el fin de mejorar la comunicación entre los involucrados. El LEL se valida con los usuarios, aplicando un chequeo informal.

Estrada et al. [Estrada 01] presentan una estrategia para construir un modelo de flujo de trabajo a partir de una plantilla de requisitos organizacional y para transformar este flujo de trabajo en un esquema conceptual orientado a objetos. Construyeron una herramienta que produce una plantilla de requisitos. La herramienta permite elegir un LEL de un dominio del negocio específico y comienza haciendo preguntas que guían el discurso. Del discurso capturado por la interacción usuario-máquina, se obtiene una especificación de requisitos formal que se traduce entonces en una plantilla.

Mauco & George [Mauco 00] [Mauco 01] produjeron heurísticas para derivar especificaciones formales escritas en el lenguaje de especificación RAISE a partir de modelos del LEL y de escenarios. En particular del LEL del dominio se derivan tipos y módulos, mientras que de los escenarios se derivan funciones. La primera etapa consiste en identificar y definir tipos del dominio, localizando en el LEL símbolos del tipo sujeto y objeto, examinando sus nociones e impactos y obteniendo mayor información mediante los vínculos a otros símbolos. En una segunda etapa se definen módulos, tanto esquemas como objetos, usando los tipos definidos previamente y buscando los componentes principales del dominio en los símbolos del LEL del tipo sujeto y objeto, y luego los módulos se organizan jerárquicamente. En la tercera etapa se definen las funciones a partir de información extraída de los escenarios.

Respecto al trabajo sobre la evolución del LEL realizado por Breitman & Leite [Breitman 98] (ver sub-sección 4.3.3), en esta tesis se toma la postura que el LEL refleja en todo momento el vocabulario del UdeD, y que su

evolución se debe básicamente a dos causas: cambios genuinos del vocabulario y mejora en la comprensión del mismo por parte del grupo de desarrollo. Luego, esta evolución natural del LEL no involucra un cambio en la perspectiva de dicho glosario respecto a la fase del desarrollo de software donde se está. Sin embargo, en el trabajo de Breitman y Leite ocurre una transformación del LEL en cada sucesiva etapa del proceso de desarrollo, que implica la definición de símbolos utilizados por el grupo de desarrollo para modelar la aplicación según la etapa del proceso. Los autores sostienen que pueden rastrear hacia atrás estos símbolos que reflejan, por ejemplo, el vocabulario de diseño o el de implementación hasta llegar a los símbolos del LEL utilizados efectivamente en el UdeD. Por ende, la utilidad de estos LELs consiste en la comprensión de los términos utilizados en los modelos construidos en cada fase del proceso.

4.5. Ficha de Información Anticipada

Se debe tener presente que con gran frecuencia cierta información del futuro se filtra ya en las etapas de comprensión del vocabulario del UdeD y de comprensión del UdeD mismo. Para que dicha información no se pierda en el cúmulo de información elicitada y no modelada, se propone el uso de una ficha (ver Figura 4-11) donde se registran individualmente requerimientos o "requisitos candidatos", es decir necesidades y deseos del futuro, o simplemente la descripción de un problema planteado por los usuarios sin una solución establecida, o alguna propuesta de solución presentada "al vuelo" por los usuarios o por los ingenieros.

El objetivo de las dos primeras etapas no es obtener información del futuro, pero es muy frecuente, principalmente cuando la fuente de información son los clientes y usuarios, que éstos quieran expresar sus demandas hacia el futuro impulsados básicamente en sus necesidades y deseos actuales insatisfechos. Dado que la recolección de esta información extra es un efecto colateral de las actividades de comprensión del vocabulario y del UdeD mismo, dicha información será incompleta, parcializada, no verificada ni validada, respondiendo posiblemente al punto de vista o intereses del cliente o usuario, y deberá ser considerada como tal, es decir como un posible requisito a contemplar en el UdeD futuro. Estas Fichas de Información Anticipada serán analizadas posteriormente durante la actividad de construcción de los escenarios futuros.

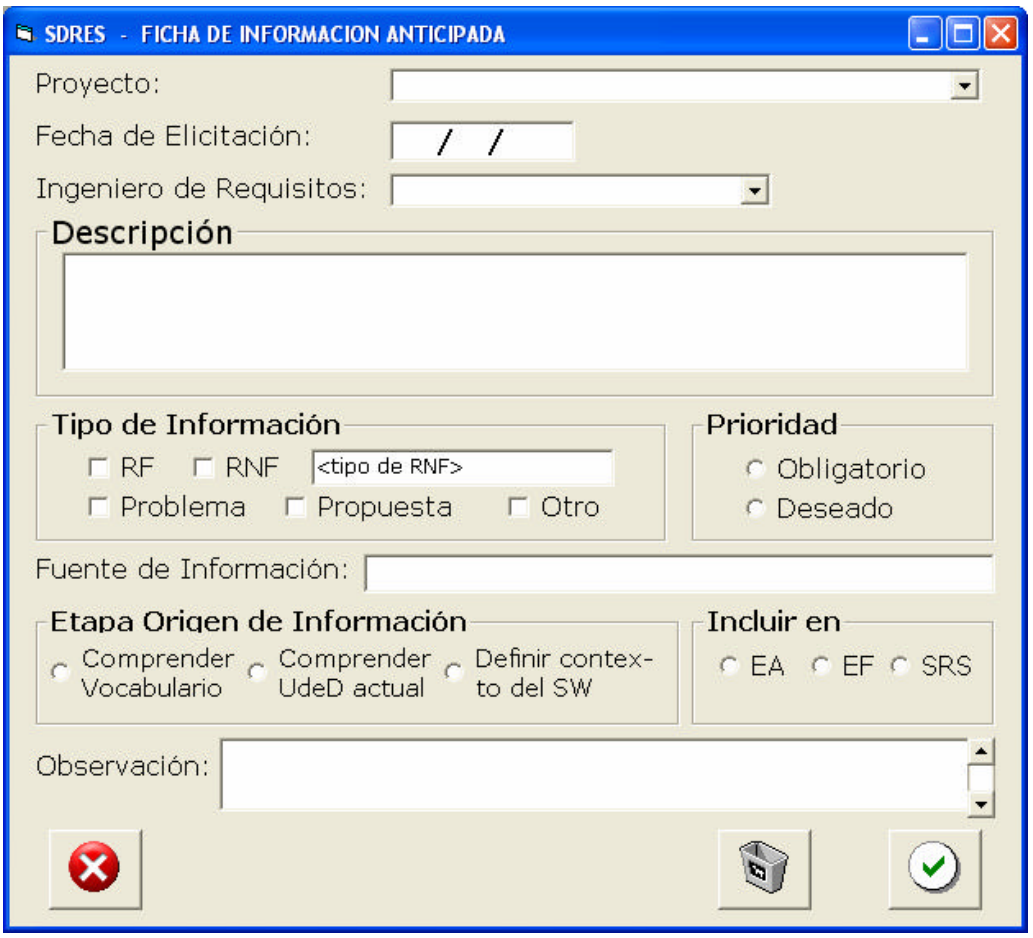


Figura 4-11. Ficha de Información Anticipada

Adicionalmente, estas fichas permiten guardar información actual, pues durante la creación del LEL es factible la elicitación de información actual que no corresponde incorporar al léxico por su nivel de detalle, pero que debe incluirse posteriormente en los escenarios actuales.

4.6. Resumen

En este capítulo se ha presentado un proceso detallado para la construcción de un glosario con términos del UdeD, denominado Léxico Extendido del Lenguaje. Este glosario será un ancla para la descripción de todos los modelos a construirse en la IR, sugiriéndose su uso en las posteriores etapas del proceso de desarrollo.

Cabe observar que este proceso de creación del LEL ha sido aplicado a múltiples casos durante varios años, algunos de ellos aplicados a la industria y otros aplicados por estudiantes en casos reales como práctica en cursos de grado y postgrado. En la Tabla 4-2 se muestran a modo de ejemplo algunos de los casos conducidos por la autora de la tesis.

Año	Caso	Institución	Autores	Cantidad de Símbolos
1995	Sistema Nacional de Emisión de Pasaportes	UB	4 Investigadores	38
1996	Sistema de Agenda de Reuniones	UB	2 Investigadores	34
1999	LEL y Escenarios para los procesos de construcción del LEL y de Escenarios	UNCPBA	2 Alumnos de grado avanzados y 2 Investigadores	78
2001	Sistema Contable	UTN-FRBA	2 Alumnos de grado	50
2001	Sistema de Obstetricia	UTN-FRBA	Alumnos de grado	67
2001	Sistema medico de Adquisición de Suministros	UTN-FRBA	Alumnos de grado	47
2001	Sistema de Gestión de Pacientes Internados	UTN-FRBA	Alumnos de grado	48
2001	Sistema de Control de Calidad para Laboratorio Farmacéutico	Empresa	1 Ingeniero de software	99
2002	Sistema de Campaña de Marketing para Banco	UTN-FRBA	Alumnos de grado	39
2002	Sistema de Gestión de Documentación Electrónica del Juzgado Laboral	UTN-FRBA	3 Alumnos de grado	42
2002	Sistema de Administración de Cajeros automáticos	Empresa	1 Ingeniero de software	55
2003	Sistema de Gestión de Colegio Secundario	UTN-FRBA	Alumnos de grado	36
2003	Sistema de Distribución de Droguería Farmacéutica	UTN-FRBA	Alumnos de grado	31
2004	Planificación y Seguimiento en un Laboratorio Farmacéutico	UTN-FRBA	3 Alumnos de grado	43
2004	Sistema de Service Técnico	UTN-FRBA	Alumnos de grado	36
2004	Sistema de Planificación y Seguimiento en un Laboratorio Farmacéutico	UTN-FRBA	3 Alumnos de grado	43
2005	Administración del Almacén de MP y artículos de Encuadernación	UNLaM	3 Alumnos de grado	38
2005	Seguimiento de Service de equipos para radiocomunicación	UNLaM	2 Alumnos de grado	37
2005	Sistema de Clasificados de un diario	UTN-FRBA	2 Alumnos de grado	33
2005	Gestión de Producción de Químicos	UTN-FRBA	2 Alumnos de grado	31
2005	Administración de Soportes con contenidos de programación por cable	UTN-FRBA	2 Alumnos de grado	31
2005	Servicio de Ecodoppler y Ecocardiografía	UTN-FRBA	3 Alumnos de grado	33
2005	Seguimiento de Producción de Cajas de Cartón – Versión 1	UNLaM	2 Alumnos de grado	41
2006	Seguimiento de Producción de Cajas de Cartón – Versión 2	UNCPBA	1 Alumno de grado	60
2006	Administración de Préstamos	UNLaM	3 Alumnos de grado	35

Tabla 4-2. Datos de casos de estudio utilizando el LEL

Como se ha presentado en la sub-sección 4.4.2, el uso del LEL ha ido más a allá de su aplicación para un proceso específico de la IR. La autora de esta tesis junto con los profesores Gladys Kaplan y Jorge Doorn han implementado su uso en un curso de grado avanzado para que los alumnos adquieran conocimiento sobre un dado modelo de proceso de software, construyendo para ello un LEL que describe el modelo a través de los términos involucrados.

En la Figura 4-12 se establece la relación entre las actividades de la IR (descriptas en la sub-sección 2.2.3) y las actividades que se realizan para la construcción del LEL. En ella se observa que Planear es básicamente una actividad de elicitación; Recolectar Símbolos involucra elicitar información y modelar mediante la construcción de una lista de símbolos; Describir símbolos es una actividad eminentemente de modelado pero que también involucra la elicitación para completar la definición de los símbolos; y Verificar y Validar son actividades netamente de análisis, pudiéndose decir que al validar siempre hay una parte de elicitación como efecto colateral. La evolución del LEL es manejada por la gestión de cambios, ya que éste es un modelo perteneciente al CORB.

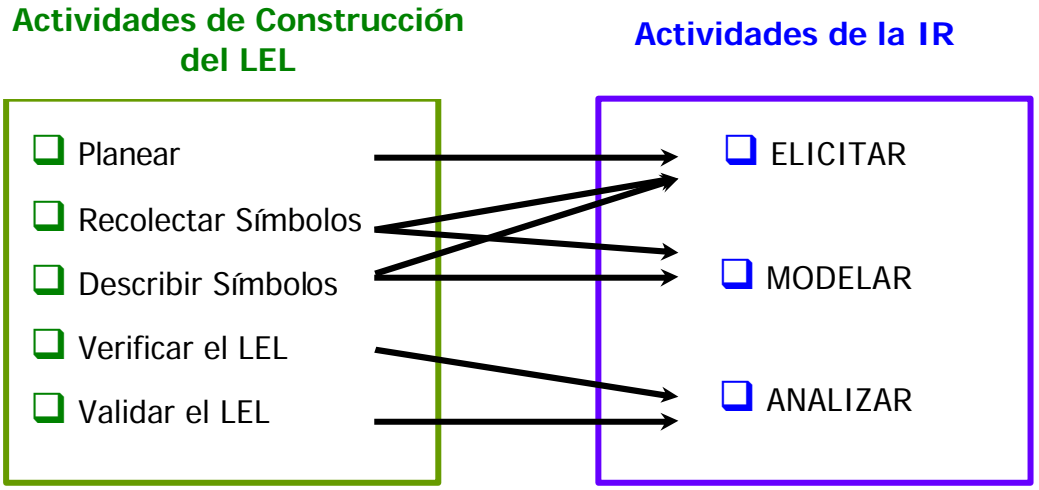


Figura 4-12. Relación entre actividades de IR y actividades del LEL

Parte del material incluido en este capítulo ha sido publicado en [Hadad 99] [Leite 04b] [Kaplan 00] [Hadad 07] [Kaplan 07].

Capítulo 5
Escenarios

«Life is Made of Situations»

American Express advertising campaign in the 90's

5.1. Introducción

Repitiendo la frase final de la sub-sección 1.1.3, los ingenieros de requisitos deben comprender, modelar y analizar el UdeD donde el software correrá y los usuarios deben confirmar que la visión de los ingenieros es correcta. Para lograr esto se requiere una técnica de comunicación atractiva para todos los participantes. Es aquí donde los escenarios se vuelven importantes, dado que pueden mantener mucha información de una forma que todos pueden utilizar.

Los escenarios son narrativas estructuradas de situaciones del macrosistema. Al igual que en el caso del LEL, los escenarios se construyen utilizando el LN con un mínimo de formalismo. Este formalismo limita y ordena el uso del LN de una manera que exige un cierto esfuerzo de los ingenieros de requisitos en su construcción pero que no limita la lectura y comprensión por parte de los clientes y usuarios, reduciendo de esta manera en forma sensible la brecha semántica entre ambos grupos humanos.

Si los escenarios se describen no sólo en LN sino intensificando el uso de los símbolos del LEL, se facilita ampliamente la validación de los escenarios y se posibilita que los símbolos sean un vínculo natural entre ambas representaciones, una característica fundamental del concepto CORB [Leite 97].

Los escenarios se usan para entender la aplicación y su funcionalidad: cada escenario describe una *situación* específica de la aplicación [Zorman 95], centrando la atención en su comportamiento. Aunque cada escenario describe una situación particular, ninguno de ellos es completamente independiente del resto de los escenarios, sino que por el contrario cada uno guarda una relación semántica con los otros [Booch 94].

Cabe aclarar el concepto de *situación*. Para considerar una situación como tal, ésta debe cumplir con las siguientes características [Breitman 01]: tener un fin establecido, poder identificar actores y recursos para satisfacer el objetivo de la situación, ocurrir en un tiempo dado continuo (sin interrupciones), tener un contexto geográfico establecido, poder ser restringida por ciertas condiciones, ser independiente (comprensible aisladamente de otras

situaciones aunque relacionada con ellas), ser concreta (anclada en la realidad) y poder presentar cursos alternativos de acción.

Los escenarios no intentan reemplazar la especificación de requisitos, son una fuente de provisión de conocimiento donde se pueden encontrar los requisitos y de donde pueden derivarse las especificaciones.

Los escenarios cumplen diferentes objetivos dependiendo de la fase del proceso de desarrollo donde se usen. En el proceso de la IR, sus objetivos son:

- ⇨ facilitar la captura de requisitos;
- ⇨ facilitar la captura de conocimiento del UdeD;
- ⇨ proporcionar un medio de comunicación entre los involucrados; y
- ⇨ mantener un ancla para la rastreabilidad.

Considerando el marco de clasificación histórica de escenarios propuesto por el marco de trabajo de CREWS [Rolland 98b], se puede decir que, según la ***vista forma***, los escenarios del CORB presentan una *notación semiformal* con vínculos de hipertexto. Con respecto a la ***vista contenido***, los escenarios son *escenarios tipo* porque se tratan a los actores y eventos por sus nombres genéricos (extraídos del LEL) y no por ejemplos particulares, además capturando el *contexto organizacional* y cubriendo todos los *aspectos funcionales* (*estructura*, *función* y *comportamiento*) y muchos *aspectos no funcionales*. Desde la ***vista propósito*** son *escenarios descriptivos*, dado que se usan para describir situaciones actuales y futuras. Finalmente, desde la ***vista ciclo de vida***, son escenarios *persistentes* pues evolucionan a lo largo del proceso de software. Los escenarios actuales se derivan del LEL mientras que los escenarios futuros se derivan de los escenarios existentes y/o de una perspectiva orientada a los objetivos. Además, la estrategia de escenarios contempla operaciones de *integración*, *refinamiento* y *expansión*, según este marco de trabajo.

Considerando la clasificación de casos de uso difundida por Alistair Cockburn en [Cockburn 00], se puede decir que los escenarios del CORB utilizan un formato "completamente vestido"[71] y, según su propósito, los escenarios actuales se corresponden con casos de uso del negocio mientras que los escenarios futuros se corresponden con casos de uso del sistema. Desde el punto de vista del nivel de detalle que muestran, se puede decir que los escenarios son de nivel objetivo de usuario, los sub-escenarios en general son de nivel sub-función y los escenarios integradores son de nivel resumen.

En SDRES, los escenarios no se crean de las mentes de los ingenieros de requisitos; su definición debe establecerse a partir de situaciones reales pues los escenarios se anclan en el contexto organizacional.

El modelo de escenarios usado en SDRES tiene una organización de descomposición, pero esto no implica que se use un enfoque top-down para

[71] Cockburn en [Cockburn 00] distingue entre casos de uso casuales y casos de uso completamente vestidos, es decir, aquellos con formato libre frente a aquellos que se ajustan a una plantilla predefinida.

construir los escenarios, ni tampoco un enfoque bottom-up. Un verdadero proceso bottom-up empezaría descubriendo episodios, luego construiría sub-escenarios y finalmente los integraría en escenarios. Un proceso top-down empezaría construyendo uno o unos pocos escenarios que abarquen todo el sistema, luego refinándolos para obtener una sucesión de conjuntos de escenarios con niveles crecientes de detalle.

En la siguiente sección se manifiesta qué se entiende en esta tesis por escenarios actuales y escenarios futuros. En la sección 5.3 se describe la representación de escenarios utilizada tanto para describir escenarios actuales como escenarios futuros, detallando sus componentes y sintaxis. En el capítulo 6 se describe en detalle el proceso de construcción de los escenarios actuales, mientras que la descripción del proceso de construcción de los escenarios futuros se desarrolla en el capítulo 7.

5.2. Escenarios Actuales y Escenarios Futuros

Los escenarios actuales y los escenarios del futuro no pueden distinguirse por su estructura, simplemente pueden diferenciarse por su contenido. Cada pieza de información en un escenario actual puede trazarse a situaciones que, en este momento, tienen lugar en el UdeD [Carroll 95], mientras que los escenarios futuros factorizan la expectativa presente en el UdeD acerca del logro del proyecto de software. En este sentido, se espera que los escenarios evolucionen de un contexto inicial a uno en prospectiva. Es más aún, las expectativas futuras cambian con el tiempo y los escenarios futuros deben reflejar estos cambios.

Debe notarse que los escenarios actuales podrían contener algunos requisitos del software pero los escenarios futuros contienen aquéllos y muchos otros nuevos elicitados, definidos y negociados durante el proceso de planificación acerca de cómo será el UdeD.

También puede decirse que hay algo así como un UdeD futuro. Debe prestarse especial atención a la palabra futuro en este contexto, dado que cualquier información registrada en un escenario futuro o supuesta perteneciente al UdeD futuro es realmente conocida en la actualidad. Cualquier referencia al futuro debe entenderse como un “futuro esperado”. Esto no implica falta de exactitud, precisión o calidad. Significa que el escenario futuro modela cosas que no existen aún, son sólo planeadas y, por supuesto, evolucionarán a lo largo del eje “tiempo de desarrollo” del CORB.

Los actores en los escenarios actuales son principalmente sujetos del LEL o personas identificables en el UdeD. Dado que no es razonable hoy en día creer que se encontrará un ambiente sin ningún sistema de software y que se construirá un software totalmente nuevo, los sistemas de software existentes también serán parte de los escenarios actuales. Sin embargo, en la visión del futuro, el sistema de software planificado será un actor principal.

Entender el UdeD actual significa comprender lo que está pasando en él. Mientras que entender el UdeD futuro significa elicitar el conocimiento de lo que se necesita o se espera. Claramente, los escenarios futuros se relacionan estrechamente con los requisitos, mientras que los escenarios actuales competen más con el conocimiento del contexto.

En esta tesis, se enfatiza la importancia del LEL y de los escenarios actuales pues son básicos a la trazabilidad. La gestión de requisitos es imposible sin la apropiada trazabilidad (ver sub-sección 2.2.7). La disponibilidad de descripciones de situaciones y la capacidad de rastrear hacia adelante y hacia atrás en la evolución de los artefactos es esencial para proporcionar un proceso de requisitos orientado a la calidad.

5.3. El Modelo de Escenario

El modelo de escenario es una estructura compuesta por las siguientes entidades: *título*, *objetivo*, *contexto*, *recursos*, *actores*[72], *episodios* y *excepciones*, y el atributo *restricción*. Actores y recursos son dos componentes enumerativos. El título, el objetivo, el contexto y las excepciones son componentes declarativos, mientras que los episodios son un conjunto de sentencias en un lenguaje simple que dan una descripción operacional de comportamiento.

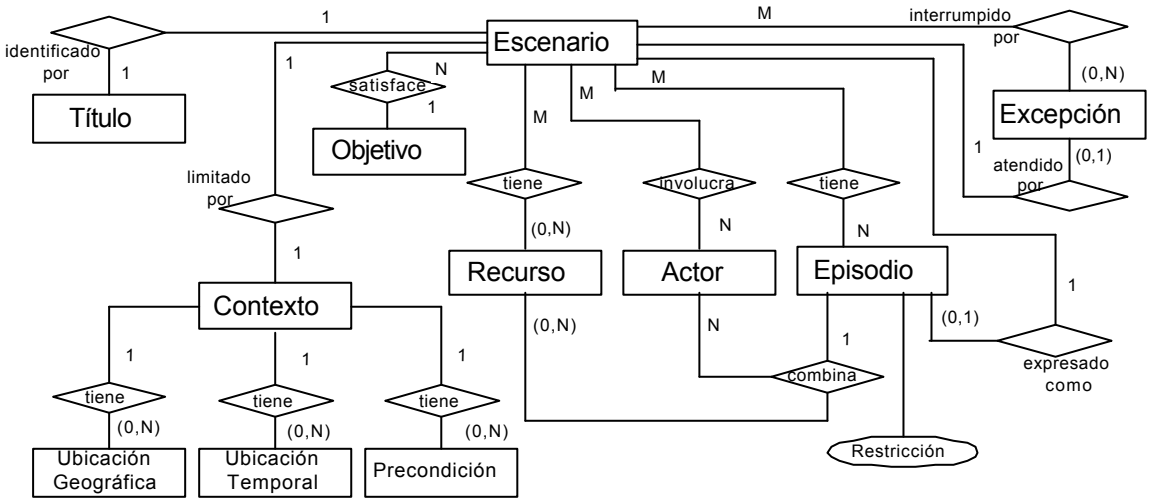


Figura 5-1. Diagrama de Entidad-Relación del Modelo de Escenario

La Figura 5-1 describe la estructura de un escenario usando el DER, donde para el componente Episodios sólo se muestran las relaciones más relevantes con otras entidades. Este componente se detalla por separado mediante un DER Extendido en la Figura 5-2.

[72] Constantine [Constantine 98] hace uso del término *Rol del Usuario* con el mismo significado.

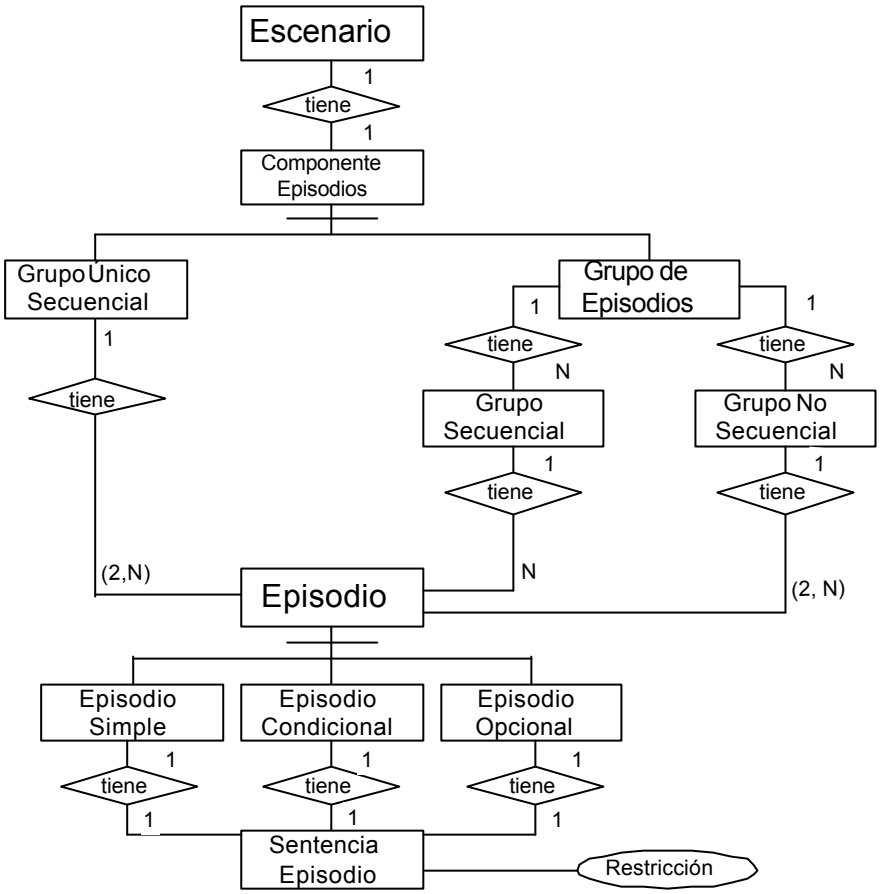


Figura 5-2. Diagrama de Entidad-Relación Extendido del Componente Episodios

La Figura 5-3 muestra la plantilla para describir escenarios basados en LN, donde se detalla la sintaxis de cada componente. El modelo de escenario presentado en esta tesis es una actualización de la versión expuesta en [Leite 00] [Leite 04b], la cual a su vez ya representaba una mejora a la versión original en [Leite 97] y encontrada también en [Hadad 97] y [Hadad 99].

El modelo de escenario debe interpretarse como pautas sintácticas y estructurales con el fin de:

- ⇨ obtener un estilo de descripción homogéneo entre el conjunto de escenarios;
- ⇨ demostrar los varios aspectos que los escenarios pueden cubrir; y
- ⇨ facilitar la verificación del escenario (principalmente mediante un proceso automatizado).

Un escenario debe satisfacer un *objetivo* que se alcanza realizando sus *episodios*. Los episodios representan el curso principal de acción pero también incluyen variaciones o posibles alternativas. Mientras se realizan los episodios puede ocurrir una *excepción*, indicando un obstáculo en el logro del objetivo del escenario.

Escenario: descripción de una situación que ocurre en el contexto del problema.
Sintaxis: Título + Objetivo + Contexto + $\{Recursos\}_1^N$ + $\{Actores\}_1^N$ + $\{Episodios\}_2^N$ + {Excepciones}

Título: identificación del Escenario. En el caso de un sub-Escenario, el título es el mismo que la sentencia episodio, sin las restricciones.
Sintaxis: Frase | ([Actor | Recurso] + Verbo + Predicado)

Objetivo: finalidad a ser alcanzada. El Escenario describe la forma de lograr el objetivo.
Sintaxis: [Actor | Recurso] + Verbo + Predicado

Contexto: compuesto por al menos uno de los siguientes ítems:
Ubicación Geográfica: lugar físico donde se produce el Escenario.
Ubicación Temporal: especificación de tiempo para el desarrollo del Escenario.
Precondición: estado inicial del Escenario.
Sintaxis: {Ubicación Geográfica} + {Ubicación Temporal} + {Precondición}
donde Ubicación Geográfica es: Frase | Nombre
donde Ubicación Temporal es: Frase | Nombre
donde Precondición es: [Sujeto | Actor | Recurso] + Verbo + Predicado

Recursos: elementos físicos o información relevantes que deben estar disponibles en el Escenario.
Sintaxis: Nombre

Actores: personas, dispositivos o estructuras organizacionales que tienen un rol en el Escenario.
Sintaxis: Nombre

Episodios: conjunto de acciones que detallan al Escenario y proveen su comportamiento. Un episodio también puede ser descripto como un Escenario.
Sintaxis (usando BNF parcial):
<episodios> ::= <serie grupo> | <serie episodios>
<serie grupo> ::= <grupo> <grupo> | < grupo no secuencial> | <serie grupo> <grupo>
<grupo> ::= <grupo secuencial> | < grupo no secuencial>
<grupo secuencial> ::= <sentencia básica> | <grupo secuencial > <sentencia básica>
<grupo no secuencial> ::= # <serie episodios> #
<serie episodios> ::= <sentencia básica> <sentencia básica> | <serie episodios> <sentencia básica>
<sentencia básica> ::= <sentencia simple> | <sentencia condicional> | <sentencia optativa>
<sentencia simple> ::= <sentencia episodio> CR
<sentencia condicional> ::= **Si** <condición> **entonces** <sentencia episodio> CR
<sentencia optativa>::= **[** <sentencia episodio> **]** CR
donde <sentencia episodio> se describe:
(([Actor] + Verbo + Predicado) | ([Actor] + [Verbo] +Título)) + {Restricción}

donde **Restricción**: es un requisito de calidad o alcance referido a una dada entidad, y se describe:
([Sujeto | Actor | Recurso] + [**No**] **Debe** + Verbo + Predicado) | Frase

Excepciones: generalmente reflejan la falta o mal funcionamiento de un recurso. Una excepción impide el cumplimiento del objetivo del Escenario. Se puede indicar el tratamiento de la excepción a través de un Escenario o de una acción simple.
Sintaxis: Causa **(**Solución**)**
donde Causa es:
(Frase | ([Sujeto | Actor | Recurso] + Verbo + Predicado)) + {Nro. Episodio}
donde Solución es:
Título | ([Sujeto | Actor | Recurso] + Verbo + Predicado)

+ significa composición, {x} significa cero o más ocurrencias de x, () para agrupar, | para or, [x] significa que x es opcional.

Figura 5-3. Modelo de Escenario

El *objetivo* del escenario se describe desde el punto de vista del negocio, y no de algún interés en particular. Esto mismo se aplica al nombre del

escenario, dado por el componente *título*. Por ejemplo, en la Figura 4-1 el título del escenario, donde se muestra la situación de un solicitante que paga un trámite, podría ser "Pagar el trámite", pero es preferible mostrar el punto de vista del sistema bajo estudio "Cobrar el trámite".

El *contexto* se describe detallando la ubicación geográfica, la ubicación temporal y las precondiciones. Los dos últimos sub-componentes pueden expresarse a través de una o más oraciones simples vinculadas por los conectores: "y", "o". Como la ubicación geográfica debe representar un único lugar para que el escenario represente una situación, entonces este sub-componente sólo puede expresarse combinando lugares con el conector "o". En el caso de ubicaciones o precondiciones alternativas, puede indicarse la condición por la que se da una circunstancia u otra. Al menos uno de estos tres sub-componentes debe estar presente en el contexto.

Los *episodios* pueden ser de tres tipos: simples, condicionales u opcionales. Los *episodios simples* son aquellos necesarios para concluir el desenvolvimiento del escenario. Los *episodios condicionales* son aquellos cuya ocurrencia dependen de una condición específica. La condición puede ser interna o externa al escenario. Las condiciones internas pueden deberse a ubicaciones o precondiciones alternativas (es decir, que contienen el conector "o") y a episodios previos. Los *episodios opcionales* (encerrados entre corchetes) son aquellos que pueden o no ocurrir dependiendo de condiciones que no pueden ser explicitadas. Para una mayor claridad, se recomienda numerar cada episodio en el orden en que se detalla en el componente.

Independientemente del tipo, un episodio puede ser expresado como una acción simple o puede ser concebido en sí mismo como un escenario, posibilitando entonces la descomposición del escenario en *sub-escenarios*.

Aunque dentro de un escenario se tratan cursos principales y alternativos de acción, su comprensión se facilita por el uso del LN, el manejo de situaciones bien delimitadas y el uso de sub-escenarios.

Un *sub-escenario* se usa cuando:

- ⇨ se detecta un comportamiento común en varios escenarios;
- ⇨ aparece un curso de acción condicional o alternativo que es complejo en un escenario; o
- ⇨ se detecta la necesidad de resaltar una situación con un objetivo concreto y preciso dentro de un escenario.

El modelo de escenario provee la descripción de comportamientos con diferentes órdenes temporales. Una secuencia de episodios implica en sí mismo un *orden de precedencia*. Para indicar un *orden no secuencial* se debe agrupar dos o más episodios utilizando el carácter numeral al comienzo y fin del grupo. Esta sintaxis se utiliza para expresar paralelismo u orden secuencial indistinto.

El atributo *restricción* se puede aplicar individualmente a los episodios, y

se utiliza para caracterizar limitaciones o condiciones de calidad respecto a la realización del episodio, habitualmente estas restricciones se asocian a RNF.

Un escenario puede ser interrumpido por *excepciones*. Cada excepción se describe con una sentencia simple que especifica la causa de la interrupción, seguido de la lista de números de episodios donde la excepción puede ocurrir. Si no se especifica una lista, implica que la excepción puede ocurrir en cualquier momento durante el desenvolvimiento de los episodios del escenario. La excepción además debe tener un tratamiento, que puede o no cumplir con el objetivo original del escenario. Cuando la excepción tiene un tratamiento especial, éste se describe en otro escenario y en el componente Excepciones se incluye entre paréntesis sólo el título de dicho escenario. Cuando la excepción tiene un tratamiento simple, se puede describir en una oración siguiendo el estilo de un episodio simple.

La Figura 4-1 muestra dos ejemplos de escenarios, donde el escenario "Trámite de Pasaporte Original" tiene dos menciones a sub-escenarios en sus episodios, siendo uno de ellos el otro escenario de la Figura "Cobrar el Trámite".

Como se ha mencionado más arriba, existe una relación jerárquica entre los escenarios, establecida mediante episodios que son en sí mismos escenarios. Los sub-escenarios surgen cuando al descubrir una situación inmersa dentro de otra se prefiere detallar a la primera en un escenario separado, probablemente con mayor nivel de detalle que en el escenario que la contiene. Así mismo esta relación se utiliza para construir escenarios integradores, los cuales agrupan escenarios temporalmente relacionados, permitiendo lograr una visión global del UdeD.

Durante el proceso de construcción de los escenarios, se suele mantener en cada escenario un componente transitorio Dudas de texto libre, que se elimina antes de finalizar el proceso, pues debe quedar vacío previamente.

5.4. Observaciones

Tanto los escenarios como los casos de uso pueden escribirse informalmente como una narrativa o utilizando alguna guía o plantilla para la estructura de los mismos. En [Jarke 98b, pág.162] se puntualiza que los practicantes exigen pautas para escribir escenarios con un texto estructurado. La narrativa simple se ha dejado en algunos casos como una primera aproximación o borrador inicial.

En concordancia con Rolland [Rolland 98c], el uso de LN en escenarios implica un modelo fácil de comprender aunque puede incluir ambigüedades e inconsistencias. Estas debilidades pueden reducirse empleando un vocabulario del UdeD bien definido, tal como el LEL, complementado con un proceso de verificación, tal como el que se presenta en la sub-sección 6.2.3.

En la clasificación presentada por [Rolland 98] ya se mostraba una gran variedad de formatos para escribir escenarios y casos de uso según distintos autores. Cockburn en [Cockburn 00] presenta una plantilla para casos de uso, con un gran número de componentes, algunos opcionales, que pueden compararse con la plantilla utilizada en esta tesis. Así mismo muestra las plantillas que utilizan otros autores:

- ⇨ Tabla de una columna (similar a la plantilla de Cockburn pero dentro de una tabla).
- ⇨ Tabla de dos columnas ideada por Rebecca Wirfs-Brock como conversaciones entre actor y sistema.
- ⇨ El estilo RUP (muy similar a la plantilla de Cockburn pero numerando cada componente).
- ⇨ El estilo OCCAM basado en un lenguaje formal desarrollado por T. Hoare.

En cuanto a los escenarios, distintos autores han creado sus propias plantillas, como [Potts 95], [Schneider 98], [Alspaugh 99], [Gough 95], [Jarke 98b].

Cabe notar entonces la gran diversidad de formatos en la estructura de escenarios y casos de uso, que puede confundir a las personas en un proyecto en cuanto a discernir sobre la elección más adecuada. Actualmente la herramienta Requisite Pro de Rational Software (http://www.rational.com/) se ha popularizado comercialmente y casi ha impuesto el uso de su plantilla en el mercado.

Hay una ambigüedad en la que estamos inmersos (ver sub-sección 2.8.1):

- para algunos autores Casos de Uso = Escenarios
- para otros autores Casos de Uso ≠ Escenarios

Cockburn y Requisite Pro caen en el segundo caso, donde el caso de uso abarca un grupo de variantes o escenarios. Otra cuestión a tener en cuenta es si el caso de uso o el escenario describe lo que se observa o lo que se espera con el sistema de software, a veces este punto de vista temporal no es mencionado. Otro punto importante es si el caso de uso o el escenario conserva el punto de vista del proceso del negocio y ve al sistema "de lejos" o por el contrario no mantiene el punto de vista del proceso del negocio y ve al sistema "de cerca".

Se debe enfatizar que para esta tesis los escenarios describen situaciones en el macrosistema (ve al sistema "de lejos") y no sólo una secuencia de interacciones entre un usuario y un sistema (ve al sistema "de cerca"), sean éstos escenarios actuales o futuros.

Los procesos de construcción de escenarios actuales y escenarios futuros (ver capítulos 6 y 7 respectivamente) están basados en un modelo de representación (Figura 5-3), pero claramente estos procesos pueden independizarse de él, adaptándose a otros modelos. La autora de esta tesis

adaptó ambos procesos al modelo de casos de uso propuesto por [Cockburn 01] y fue aplicado en el curso de "Ingeniería de Requerimientos" en la Carrera de Especialización en Ingeniería de Software del postgrado de la UCA en el 2006. Ver Apéndice G sobre la correlación entre ambos modelos.

El trabajo en evolución de escenarios presentado en [Leite 97] [Breitman 98] [Breitman 05] no ha enfatizado una división entre escenarios actuales y escenarios futuros. Sin embargo resaltar las diferencias entre estos tipos de escenarios ayuda al aspecto ingenieril de construir los escenarios.

En el trabajo de [Breitman 05] una perspectiva de evolución se presenta por el avance en el proceso de desarrollo, considerando escenarios de especificación, escenarios de diseño y escenarios de implementación. Comparando con SDRES, los escenarios de especificación se corresponderían con los escenarios futuros. Los escenarios de diseño y de implementación estarían fuera del alcance de la estrategia, pero podrían ser trazables como modelos externos al CORB.

5.5. Resumen

Se ha presentado un modelo de escenario que se utiliza para representar tanto situaciones actuales que ocurren en el UdeD como situaciones esperadas con el nuevo sistema de software, haciendo hincapié en el contexto organizacional en el que se desarrollan los procesos del negocio. Este modelo de escenario permite representar todas las variantes posibles de una situación, incluyendo casos alternativos y excepciones, permitiendo vincular restricciones a las actividades (episodios) que se desarrollan. Se encuadra cada situación en un contexto temporal, geográfico y de estado inicial, indicando los recursos necesarios y los actores involucrados.

Parte del material incluido en este capítulo ha sido publicado en [Leite 00] [Doorn 02] [Leite 04b].

Capítulo 6
Escenarios Actuales

Father Brown laid down his cigar and said carefully:
«It isn't that they can't see the solution.
It is that they can't see the problem. »

G.K. Chesterton, "The Point of a Pin" en The Father Brown Stories, Cassell & Co, Londres 1929.

6.1. El proceso de construcción de escenarios actuales

6.1.1. Generalidades del proceso

El proceso de construcción de los EA parte del léxico del dominio de la aplicación pues el LEL contiene información que puede ser utilizada en los escenarios, produciendo entonces una primera versión de los escenarios derivados exclusivamente del LEL. Estos EA son completados y mejorados elicitando información de otras fuentes, y posteriormente organizados con el fin de obtener un conjunto consistente de escenarios que representen el dominio de la aplicación. Durante o después de estas actividades, los escenarios son verificados y validados para detectar Discrepancias, Errores y Omisiones.

El proceso se describe en la Figura 6-1 usando el modelo SADT, donde se pueden observar las siguientes actividades: 1) *Derivar*, 2) *Describir*, 3) *Organizar*, 4) *Verificar* y 5) *Validar*.

Aunque el proceso graficado en la Figura 6-1 muestra un flujo principal compuesto de tres actividades: *Derivar*, *Describir* y *Organizar*, éstas no son estrictamente secuenciales. Algunas actividades se realizan concurrentemente como condición intrínseca del proceso mismo, siempre presente en estos casos [Goguen 93]. Adicionalmente, existe un mecanismo de retroalimentación entre ellas, que aporta una segunda instancia de paralelismo. Esto ocurre cuando se realizan las actividades *Verificar* y *Validar*, retornando a la actividad Describir, donde las correcciones son realizadas en base a las listas de DEOs[73]. La actividad *Verificar* ocurre después de describir los escenarios y también después de organizarlos, cuando aparecen nuevos escenarios o se generan los escenarios integradores. Los escenarios son validados con los usuarios después de la verificación.

Se puede producir una tercera lista de DEOs, que actúa como retroalimentación al proceso de construcción del LEL, para mantener la consistencia entre el vocabulario de los escenarios y el LEL mismo. Esto, en

[73] Esta es una lista de DEO similar a la generada en el proceso de construcción del LEL pero referida a defectos descubiertos en los escenarios.

general, no ocurre porque cambió el vocabulario del UdeD sino debido a que aumenta el conocimiento sobre dicho vocabulario durante la construcción de escenarios y se debe entonces rectificar el LEL. Incluso se pueden generar nuevos términos, no a causa de su uso en los escenarios, sino porque realmente son parte del vocabulario del UdeD y que no habían sido detectados en la etapa anterior de SDRES.

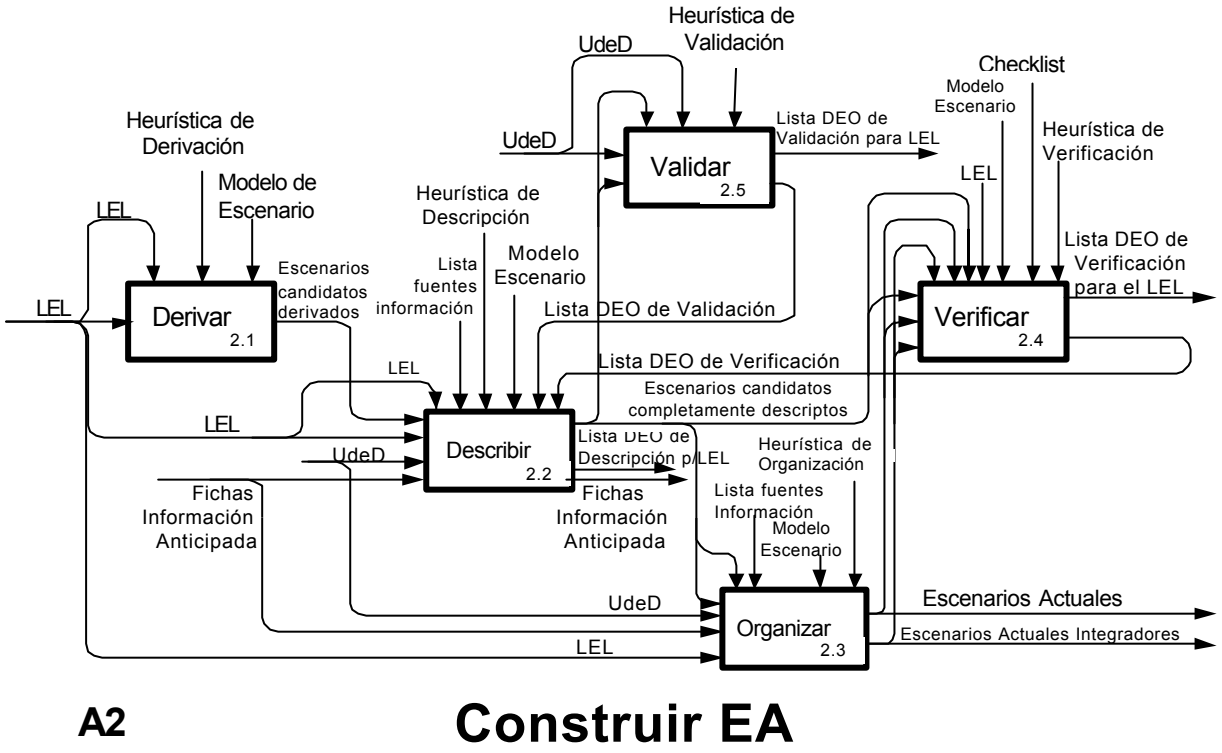


Figura 6-1. SADT del proceso de construcción de Escenarios Actuales

La siguiente fase de SDRES, Definir el contexto del software, también puede producir una lista de DEO sugiriendo cambios y agregados en el conjunto de EA.

6.1.2. Actividades del proceso

A continuación se detallan las actividades graficadas en el SADT de la Figura 6-1, y para las actividades *Derivar* y *Organizar*, se incluye más detalle presentando un modelo de SADT para la descomposición de cada uno de ellas, ver las Figuras 6-2 y 6-5 respectivamente.

(1) DERIVAR

Esta actividad tiene como propósito generar escenarios candidatos utilizando información contenida en el LEL, basándose en el modelo de escenario y aplicando las heurísticas apropiadas. El proceso de derivación consiste en tres pasos: identificar los actores del UdeD, identificar escenarios candidatos y por último crearlos extrayendo información almacenada en el LEL. La Figura 6-2 muestra el modelo SADT en el que se descompone la actividad *Derivar*.

(1.1) IDENTIFICAR ACTORES

Se genera una lista preliminar de actores, a partir de información contenida en el LEL, con el fin de guiar las siguientes sub-actividades.

Todo símbolo clasificado como Sujeto en el LEL se considera como un posible actor. Es esperado que esto ocurra salvo contadas ocasiones, como puede ser el caso de actores off-stage, es decir, aquellos que son mencionados en el curso de acción del escenario pero que no interactúan directamente con otros actores. Es también factible que algún símbolo del tipo Objeto en el LEL pueda tratarse como un actor en un escenario, ejemplo de ello puede ser un sistema de software, una máquina o un sector de la organización, donde quisiera mostrarse su funcionamiento.

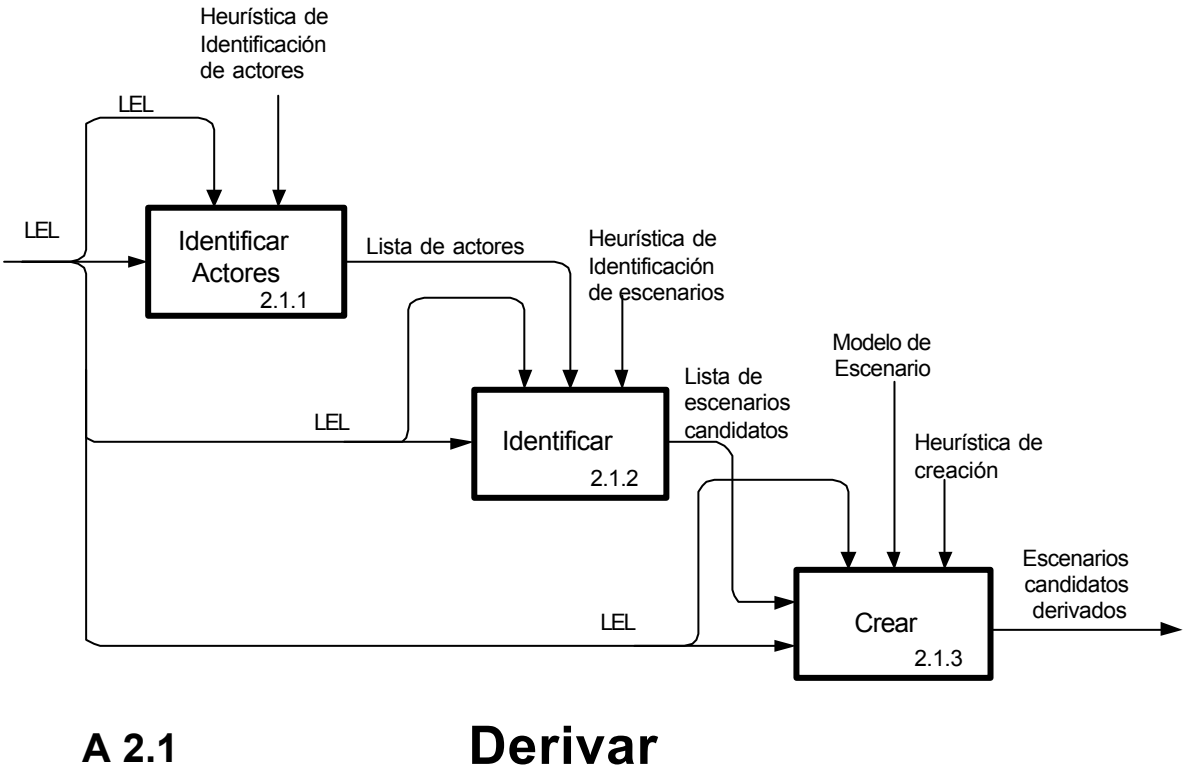


Figura 6-2. SADT de la actividad Derivar en el proceso de EA

Se ha detectado en la práctica con cierto grado de frecuencia, que dos o más Sujetos del LEL comparten una responsabilidad o pueden realizar la misma actividad. Aquí caben dos tratamientos posibles para su conversión en Actores de escenarios: i) crear un rol[74] que asuma la responsabilidad de esta actividad, o ii) utilizar en el escenario correspondiente la lista de Sujetos que realizan dicha actividad. La primera alternativa involucra la creación de un nombre de Actor, mientras que la segunda involucra tener como Actor una lista de Sujetos alternativos. La decisión enfrenta al dilema legibilidad versus facilidad de escribir. Como es un aspecto muy analizado en los lenguajes de programación, a mayor legibilidad se tendrá menor escribibilidad, siendo la solución clásica en ese ámbito una compensación que no maximice ninguna de ambas cualidades. En concreto, la solución depende del problema pues las dos propiedades son totalmente válidas. En el caso particular de escenarios, se recomienda usar la creación de un rol si el término utilizado está disponible en el UdeD o es fácilmente reconocido por los usuarios, sino será conveniente mantener la lista de actores alternativos que puedan cumplir dicha actividad.

Cuando el LEL involucra el lenguaje de toda o gran parte de la organización y el problema bajo estudio involucra un ámbito más acotado, entonces se deberá determinar cuáles símbolos del tipo Sujeto tienen alguna participación en dicho ámbito y armar la lista de actores con ese grupo específico de símbolos Sujeto. Esto es lo que se haría si se está siguiendo una estrategia iterativa incremental (ver sub-sección 3.4) pues se construyen los EA que corresponderían a ese incremento. Aún no siguiendo este tipo de estrategia, ocurre que durante la construcción del LEL no se conocen con precisión los límites del UdeD y menos aún del futuro sistema de software. A esto hay que sumarle que por su propia conectividad los símbolos del LEL suelen tener una cierta tendencia a abarcar dominios más amplios que el que se debe estudiar. En estos casos es viable la reducción de la lista de actores para centrarse en un contexto más acotado.

(1.2) IDENTIFICAR ESCENARIOS

Se extraen del LEL los impactos de los símbolos elegidos como actores. Cada impacto representa un posible escenario y, por lo tanto, se crea un título para el mismo preliminar y se lo incorpora a la lista de escenarios candidatos. El título del escenario se compone con la acción (verbo) incluida en el impacto, expresada en infinitivo, más un predicado también obtenido del impacto. Se debe descartar aquél impacto que involucra un punto de vista formal pues, según lo mencionado en la sub-sección 4.3.2, dicho impacto no representa un hecho de lo actual.

Cuando diferentes actores ejecutan una misma actividad, es muy probable que dos o más escenarios de la lista puedan compartir el título. En este caso es recomendable no eliminar ninguno hasta asegurarse en pasos

[74] Se debe tener plena conciencia que se está introduciendo un recurso de modelado que no refleja exactamente la realidad, si bien la distorsión incorporada es mínima.

siguientes que realmente son iguales; es conveniente agregar el actor al título de cada escenario similar de manera de diferenciarlos.

Es recomendable redactar el título del escenario desde el punto de vista del negocio, y no de agentes externos a la organización como clientes o proveedores, de modo tal de tener presente siempre los objetivos del negocio.

(1.3) CREAR

En este paso se intenta construir los escenarios con tanta información como sea posible extraída del LEL, aplicando las heurísticas de creación. El producto de este paso es un conjunto de escenarios candidatos derivados.

La estructura del impacto que dio origen al escenario es muy importante en cuanto a la cantidad y calidad de la información presente en el LEL que contribuye a la creación del escenario. Es así que, si el impacto mencionado contiene otros símbolos del LEL, los mismos son a su vez una fuente valiosa de información para componentes tales como el contexto, los actores y los recursos entre otros. Se debe analizar además el contenido de dicho impacto buscando símbolos del LEL del tipo Verbo.

⇨ Si existe un símbolo Verbo en el impacto:

i) El objetivo se define preliminarmente en base al título del escenario y la noción del símbolo Verbo.
ii) La ubicación geográfica, la ubicación temporal y la precondición del contexto puede extraerse de la noción del símbolo Verbo. Para la precondición, también puede haber información relevante observando semánticamente las relaciones con los restantes impactos del símbolo Sujeto origen.
iii) Los actores y recursos se identifican a partir de la información del símbolo Verbo, buscando en él símbolos del tipo Sujeto y del tipo Objeto respectivamente.
iv) Los episodios se derivan de cada impacto del símbolo Verbo.
v) En el caso de detectarse un símbolo del tipo Estado en la descripción del Verbo, existe la posibilidad de incluir como episodio alguna acción en la descripción del símbolo Estado que tuviera relación con el escenario.

⇨ Si no existe un símbolo Verbo en el impacto:

i) Se identifican otros tipos de símbolos del LEL en el impacto y se toman como posibles fuentes de información.
ii) El objetivo se define preliminarmente en base al título del escenario.
iii) La ubicación geográfica, la ubicación temporal y la precondición del contexto pueden ser extraídas del impacto del símbolo que originó el escenario (símbolo Sujeto). Para la precondición, también puede haber información relevante observando semánticamente las relaciones con los restantes impactos del mismo símbolo Sujeto.

iv) Se seleccionan posibles actores y recursos a partir de los símbolos detectados previamente, leyendo toda sus descripciones, además del símbolo Sujeto que dio origen al escenario. Los actores se derivan de símbolos tipo Sujeto y los recursos de símbolos tipo Objeto.
v) En general, no se derivan episodios desde el LEL, ellos serán detectados en las actividades siguientes.
vi) Sin embargo, existe la posibilidad de que a través de los símbolos Objeto, símbolos Estado u otros símbolos Sujeto detectados, se puedan incluir como episodio alguna acción referida a este escenario.

La Figura 6-3 ejemplifica la actividad *Derivar* usando el caso "Sistema de Agenda de Reuniones" desarrollado en 1997 durante el proyecto de investigación Uso de Escenarios en el Desarrollo de Software en la Universidad de Belgrano, y la Figura 6-4 muestra la versión final de este escenario. Las palabras o frases subrayadas son símbolos del LEL del *Sistema de Agenda de Reuniones*. En la Figura 6-3 se muestran números para explicar la actividad *Derivar*, cuyo significado se detalla a continuación:

1 - de las entradas del LEL se genera una lista de actores;
2 - seleccionando un actor se tiene la correspondiente entrada del LEL;
3 - del impacto de esta entrada del LEL se produce una lista de escenarios candidatos;
4 - seleccionado un escenario candidato se encuentra el símbolo Verbo del LEL correspondiente al título del escenario;
5 - se usa la noción del símbolo Verbo para definir el objetivo, actor y recursos;
6 - el impacto del símbolo Verbo proveerá las bases para describir los episodios; y
7 - del impacto originado en 3 (tipo Sujeto) se deriva información para el contexto.

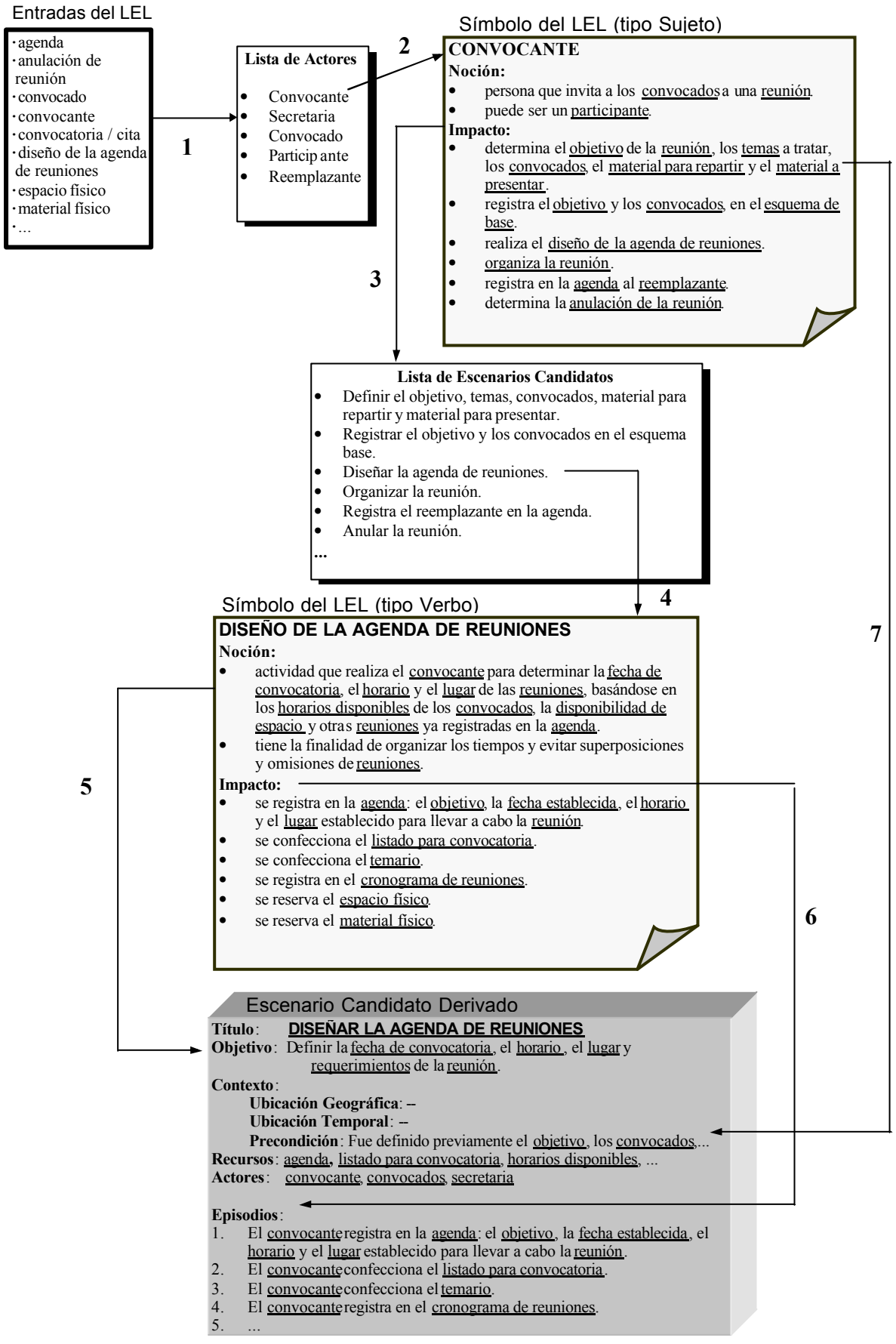


Figura 6-3. Ejemplo de derivación de un escenario actual

Sistema de Agenda de Reuniones

Título: **DISEÑAR LA AGENDA DE REUNIONES**
Objetivo: Definir la fecha de convocatoria, el horario, el lugar y requerimientos de la reunión.
Contexto:
Ubicación Geográfica: –
Ubicación Temporal: –
Precondición: Debe presentarse previamente la necesidad de una reunión.
Recursos: agenda, horarios disponibles, cronograma de reuniones, esquema de base
Actores: convocante, secretaria

Episodios:
1. **[**El convocante obtiene del esquema de base los datos de la reunión a diseñar.**]**
2. **# Si** no se tienen registrados los horarios disponibles de los convocados, SOLICITAR HORARIOS DISPONIBLES**.**
3. El convocante consulta sobre la disponibilidad de espacio.
4. El convocante consulta la agenda para ver sus horarios disponibles en la misma. #
5. ESTABLECER LA FECHA DE LA REUNION.
6. **[**El convocante define el material para repartir.**]**
7. **#**El convocante o la secretaria registra en la agenda: objetivo, temas a tratar, fecha establecida, horario, lugar, convocados, material a presentar y material para repartir.
8. GENERAR EL LISTADO PARA CONVOCATORIA.
9. **[**El convocante define el temario.**]**
10. El convocante o la secretaria registra la reunión en el cronograma de reuniones.
11. El convocante o la secretaria reserva el espacio físico.
12. **[**El convocante o la secretaria reserva el material físico.**]**
13. **[**El convocante o la secretaria registra el material físico en la agenda.**] #**

Excepciones:
Conflictos en la disponibilidad de espacio. Episodio: 11. (SOLICITAR ESPACIO FISICO EN EDIFICIO ANEXO)
Conflictos en la disponibilidad de material físico. Episodio: 12.(El convocante solicita autorización al Sector Mantenimiento para obtener material físico especial)

Figura 6-4. Versión final de un escenario actual

(2) DESCRIBIR

Esta actividad tiene por objetivo completar la descripción de los escenarios candidatos, agregando información del UdeD basándose en el modelo de escenario, utilizando en la descripción los símbolos del LEL y aplicando las heurísticas de descripción. El resultado es un conjunto de escenarios descriptos "completamente".

El conjunto incompleto de escenarios candidatos obtenidos en el paso anterior debe extenderse en dos sentidos: agregando nuevos escenarios y completando el contenido de los escenarios. Entonces, los ingenieros de requisitos retornan a las fuentes de información identificadas durante la creación del LEL para recolectar más información. Además, ahora se utiliza la información recogida durante el proceso de construcción del LEL pero que no fuera incluida en él, es decir, se utilizan aquellas Fichas de Información Anticipada que contengan información actual.

Esta actividad debe planearse y generalmente se realiza aplicando las técnicas de entrevistas estructuradas, observación y lectura de documentos,

pero en general casi todas las técnicas de elicitación son aplicables[75], lo que debe identificarse es la vigencia de la información que brinda la fuente, pues en esta etapa se requiere estrictamente información actual. Primero, la recolección de información apunta a confirmar y mejorar los cursos normales de eventos.

Después, deben descubrirse alternativas y casos de excepción. Algunas causas de excepción se elicitan de las fuentes de información, mientras que otras pueden deducirse cuestionando la ocurrencia de episodios o la falta de disponibilidad o mal funcionamiento de los recursos. Al descubrir causas de excepción, los ingenieros de requisitos deben averiguar sobre el tratamiento que recibe la excepción en el UdeD, esto puede implicar una nueva situación a ser descripta a través de un escenario separado. En la precondición del escenario excepción se incluye la causa de la excepción. Si el escenario excepción se utiliza como tratamiento de más de una excepción, la precondición puede expresarse por enunciación como una lista alternativa de causas o por generalización de dichas causas.

También deben detectarse restricciones, algunas elicitadas del UdeD y otras examinando los episodios. Ya en el UdeD actual pueden existir RNF para el software existente, que deben ser considerados e incorporados en el atributo Restricción. Una vez concluida la descripción del escenario, deben repasarse los episodios cuidadosamente para confirmar si tienen un orden secuencial o no y para detectar si algún episodio se realiza bajo alguna condición (episodio condicional) o si éste no siempre se realiza pero no hay una condición explícita que se pueda especificar (episodio optativo).

Toda cuestión dudosa que surja durante la escritura del escenario, debe volcarse en el componente transitorio Duda, para luego poder dilucidarlo en el UdeD.

Observando la Figura 6-1, se puede ver que existe un reciclo desde las actividades *Verificar* y *Validar*, representado por las listas de DEOs. Estas listas pueden motivar cambios en las descripciones de los escenarios. También la propia actividad *Describir* puede generar una lista de DEOs referidos al LEL.

A continuación se enuncian las heurísticas generales para tener en cuenta durante la descripción de los escenarios. Dado que al llegar a esta actividad, ya se cuenta con algunos escenarios parcialmente descriptos, estas guías deben usarse para mejorar las descripciones ya existentes.

- ✓ Escriba preferentemente usando oraciones cortas.
- ✓ Evite el uso de más de un verbo por oración.
- ✓ Escriba las oraciones maximizando el uso de los símbolos del LEL.
- ✓ Seleccione Actores y Recursos que sean preferentemente símbolos del LEL del tipo Sujeto y del tipo Objeto respectivamente.
- ✓ Escriba el objetivo en forma precisa y concreta.
- ✓ Debe completar al menos uno de los sub-componentes del contexto.

[75] Técnicas de elicitación como por ejemplo: análisis de dominios, reuso de requisitos o prototipos, no son técnicas que sirvan para recolectar datos referidos a lo actual.

- ✓ Incluya en el componente Recursos todos aquellos recursos que se utilizan en los episodios o son referidos implícitamente por el verbo del episodio. Ejemplo: "*El cajero timbra el formulario*", donde el recurso implícito es la *Máquina Timbradora*.
- ✓ Excluya recursos triviales del componente Recursos. Ejemplo: "*El contador firma el cheque*", donde el elemento para firmar, ya sea una lapicera o birome, no es un recurso relevante para la acción.
- ✓ Recuerde que un recurso no puede ser aquél generado durante la realización de los episodios del escenario, sí podrá ser un recurso en otro escenario. Ejemplo: "*El convocante confecciona la lista para la convocatoria*", donde *lista para la convocatoria* es un recurso en varios otros escenarios pero no en el escenario al que pertenece este episodio.
- ✓ Incluya también en el componente Recursos aquellos recursos utilizados en sub-escenarios del escenario.
- ✓ Incluya en el componente Actores aquellos involucrados en los episodios que tienen alguna participación en ellos. La mención de un actor en un episodio no significa su participación. Ejemplo: "*El contador analiza la cuenta corriente del cliente*", donde el cliente no realiza ninguna acción en este episodio, aún cuando es un actor en otros escenarios. Se dice que el cliente es un actor off-stage en este escenario.
- ✓ Incluya también en el componente Actores aquellos actores que participan en sub-escenarios del escenario.
- ✓ Utilice un verbo en cada episodio que sea preciso y concreto, especificando la acción final y evitando posibles ambigüedades.
- ✓ Asegúrese que los episodios ocurren dentro de la ubicación geográfica y temporal descripta en el Contexto del escenario.
- ✓ Describa los episodios en tiempo presente.
- ✓ Evite usar en los episodios verbos tales como: podría, puede, debe, debería.

Toda información elicitada que no corresponda volcar en los escenarios actuales pues es información sobre lo esperado para el futuro, deberá incluirse en las Fichas de Información Anticipada.

(3) ORGANIZAR

Esta actividad es una de las más complejas y sistematizadas en el proceso de construcción de escenarios. El conjunto de escenarios actuales obtenidos de la actividad *Describir* puede presentar dos tipos de debilidades: i) los ingenieros de requisitos están sumergidos en detalles, perdiendo la visión global del UdeD, y ii) existe un riesgo de inconsistencias entre los escenarios, tales como la existencia de sub-escenarios irrelevantes o escenarios con alta similitud. La primera se encara por medio del uso de escenarios integradores actuales, los cuales describen situaciones "artificiales" presentadas sólo con el propósito de hacer más entendible y manejable el conjunto de escenarios. Los escenarios integradores brindan una visión global de la aplicación. La segunda debilidad se maneja componiendo o descomponiendo escenarios, para atacar

básicamente problemas de falta de homogeneidad en la descripción de las situaciones y algunos problemas menores de semántica.

Cuando se encaran aplicaciones medianas a grandes, la cantidad de escenarios suele tornarse inmanejable. Luego, los escenarios integradores dan un panorama de las relaciones entre varias situaciones, dado que cada episodio de un escenario integrador es el título de un escenario ya descripto. Por lo tanto, el propósito principal de los escenarios integradores es vincular escenarios dispersos, proveyendo un panorama completo de la aplicación, mientras se preserva el formato de LN encarado por la estrategia.

Organizar los escenarios consiste en reorganizarlos e integrarlos, comenzando a partir de escenarios completamente descriptos. Aunque la actividad *Organizar* está ubicada como tercera caja en la Figura 6-1, las actividades *Verificar* y *Validar* deben realizarse tanto antes de comenzar a organizar los escenarios como luego de terminada esta actividad. Es decir, organizar escenarios es una tarea que se hace sólo cuando se tiene una buena confianza en contar con un conjunto de escenarios bien definidos, idealmente cuando las listas de DEOs están vacías. Luego, como consecuencia de esta actividad, se procede a una verificación y validación, afectando sólo las partes modificadas y los nuevos escenarios.

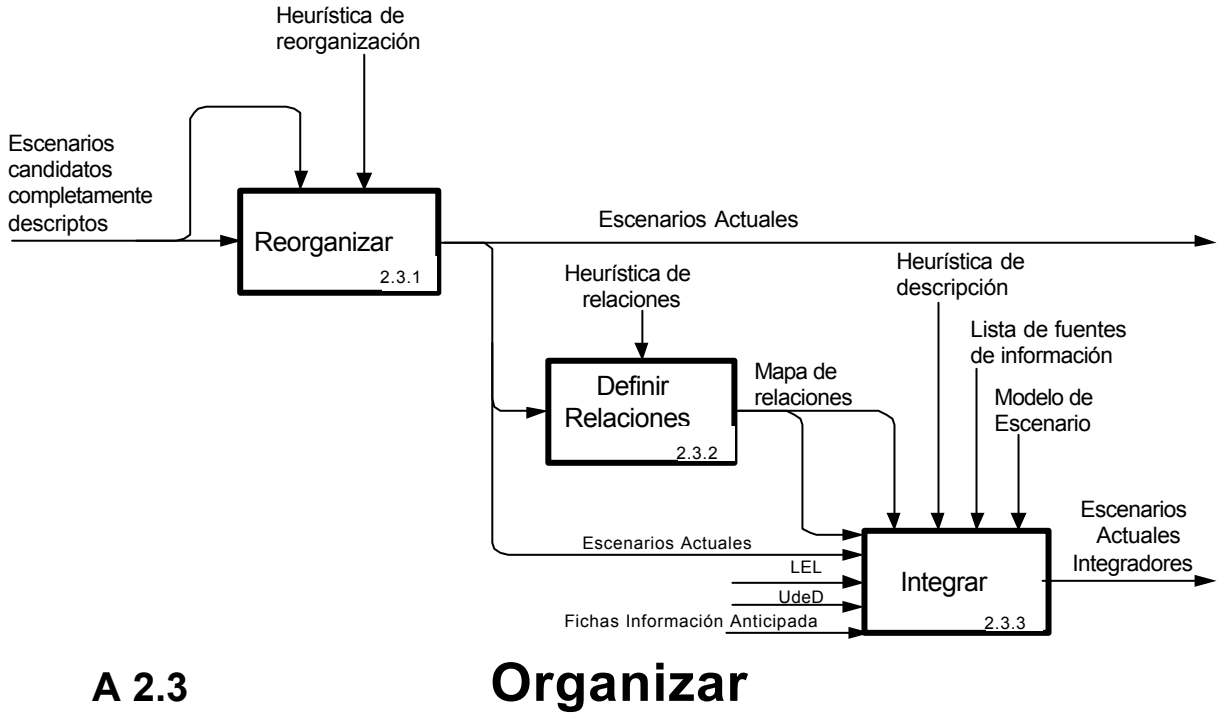


Figura 6-5. SADT de la actividad Organizar en el proceso de EA

En esta actividad puede ser necesario volver al UdeD, utilizar el LEL y chequear contra el modelo de Escenario. Los problemas mencionados pueden minimizarse utilizando las heurísticas que se detallan para la actividad *Organizar*. Dichas heurísticas utilizan una taxonomía de operaciones y relaciones. La Figura 6-5 muestra el modelo SADT que descompone la actividad *Organizar*.

(3.1) REORGANIZAR

Los escenarios pueden tener distinto grado de detalle o superposición, especialmente cuando son construidos por más de un ingeniero de requisitos. Esto se torna más complejo a medida que se tienen más escenarios y un grupo de trabajo más grande. Básicamente la reorganización de escenarios consiste en componer dos o más escenarios en uno, o en fragmentar un escenario en dos o más. La composición se realiza cuando una única situación fue artificialmente repartida en varios escenarios durante las actividades *Derivar* o *Describir*. Por otro lado, se fragmenta o descompone un escenario cuando contiene más de una situación. Estas operaciones se agrupan de a pares (composición / descomposición) bajo tres aspectos:

- ⇨ Granularidad
- ⇨ Situaciones con variaciones
- ⇨ Temporalidad

Durante las actividades de *Derivar* y *Describir* el foco no está en los sub-escenarios, por lo tanto, en este paso los sub-escenarios derivados requieren una atención especial para confirmar que ellos son realmente necesarios y que no existen sub-escenarios ocultos en otros escenarios, con el fin de clarificar el conjunto de escenarios.

En el UdeD puede haber situaciones muy similares entre sí que presentan algunas diferencias o variantes en su realización, como por ejemplo, la existencia de más una ubicación geográfica posible, más de un actor ejecutando la misma tarea, entre otros. El modo en que fueron elicitadas depende de diversos factores, tales como la cantidad de ingenieros de requisitos involucrados, el orden en que se trataron los impactos de los sujetos del LEL, entre otros.

La duración de un escenario es también una cuestión difícil de establecer. En algunos casos, dos escenarios están tan relacionados que uno se produce inmediatamente después del otro, por lo tanto, puede ser mejor describirlos en un solo escenario. Por otro lado, un escenario con un lapso de vida (contexto temporal) largo conteniendo episodios no muy compactos semánticamente pueden conducir a la creación de dos o más escenarios diferentes.

En la tabla 6-1, se enumeran las operaciones aplicables según cada aspecto.

Aspecto	Operación	Tipo
Granularidad	*Aplanar*	Composición
	Factorizar	Descomposición
Variaciones	*Consolidar*	Composición
	Dividir	Descomposición
Temporalidad	*Fusionar*	Composición
	Separar	Descomposición

Tabla 6-1. Tipos de operaciones en la reorganización de escenarios

Para establecer la necesidad de las operaciones de composición / descomposición, se deben identificar previamente algunas propiedades y relaciones de los escenarios. Cuando un escenario presenta propiedades especiales principalmente en sus episodios, puede aplicarse descomposición. Por otro lado, la composición puede practicarse cuando se descubren relaciones específicas entre los escenarios: superposición, orden de precedencia o relación jerárquica.

Debe notarse que tanto la composición como la descomposición involucran el riesgo de una degradación semántica. La descomposición crea dos o más escenarios donde había previamente uno sólo. Este procedimiento no debe crear situaciones "artificiales"; es decir, la descomposición debe aplicarse sólo cuando los escenarios resultantes describan mejor el UdeD. Por el contrario, la composición reemplaza dos o más escenarios por uno.

Las operaciones se sugieren como guías para mejorar la comprensión y administración de los escenarios. Por lo tanto, la reorganización no debe asumirse como una actividad obligatoria sino como una buena práctica.

A continuación se describen las operaciones con sus correspondientes propiedades o relaciones entre escenarios:

Aplanar[76]	
Oportunidad	Esta operación puede aplicarse cuando se detectan sub-escenarios no relevantes con poca ocurrencia en otros escenarios. Debe cumplirse adicionalmente que los episodios del sub-escenario presenten la misma granularidad que los episodios de cada escenario que lo menciona.

[76] En [Breitman 98], se definió y usó otra operación aplicada a escenarios, denominada Encapsular; realmente Aplanar y Fusionar son una especialización de Encapsular.

Procedimiento	La operación Aplanar incorpora los episodios del sub-escenario dentro de cada escenario que lo menciona. Si el sub-escenario presentaba excepciones, éstas deben trasladarse a los escenarios aplanados. El sub-escenario original se elimina cuando todas sus ocurrencias se han aplanado.
Propósito	Esta operación permite reducir la cantidad de escenarios no relevantes y, por lo tanto, facilita su administración. El conjunto resultante de escenarios decrece la profundidad de la jerarquía en todos los puntos de aplanamiento. Esta operación es análoga a la característica inline en la generación de código en algunos lenguajes de programación.

Factorizar	
Oportunidad	Esta operación puede aplicarse cuando dentro de un escenario se detecta un conjunto de episodios muy relevantes o que presentan diferente nivel de detalle respecto al resto de los episodios. Esta operación también se puede aplicar cuando se descubre la ocurrencia de un mismo conjunto de episodios en dos o más escenarios con comportamientos comunes.
Procedimiento	Factorizar crea un sub-escenario extrayendo el conjunto de episodios detectado de uno o más escenarios. El grupo de episodios en cada escenario origen se reemplaza por el título del nuevo sub-escenario que los contiene. Esta operación debería realizarse sólo si se puede encontrar un objetivo significativo para el conjunto de episodios a factorizar en el nuevo sub-escenario, sino se estaría creando una situación ficticia.
Propósito	Esta operación contribuye a que los escenarios sean más fáciles de comprender, pues todos los episodios del escenario quedan al mismo nivel de detalle. Fomenta la reusabilidad, dado que un sub-escenario posee un objetivo más concreto y limitado, con mayor posibilidad de estar presente en otras situaciones.

Consolidar	
Oportunidad	Esta operación puede aplicarse a escenarios superpuestos. Dos o más escenarios tienen una superposición fuerte si sus objetivos y contextos son similares, y tienen varias coincidencias en sus episodios. Es decir, estos escenarios presentan el mismo “núcleo de acción”.

Procedimiento	Consolidar copia los episodios comunes de los escenarios originales a un escenario nuevo y crea nuevos episodios condicionales usando los episodios no compartidos, siendo la condición parte de las precondiciones de los escenarios originales. El objetivo del nuevo escenario deberá abarcar a todos los episodios incorporados, entonces se dice que el objetivo se extiende por generalización. Los escenarios originales usualmente tienen los mismos actores y muchos recursos comunes, por ende en el nuevo escenario deberá incluirse la unión de los actores y de los recursos, como también la unión de las excepciones. Luego, los escenarios originales se eliminan.
Propósito	Esta operación permite reducir redundancia en el conjunto de escenarios, facilitando el mantenimiento de los escenarios.

Dividir	
Oportunidad	Esta operación puede aplicarse cuando se detecta la presencia de muchos episodios condicionales, dirigidos por el mismo tipo de condición. Ésta puede presentarse por la negación de la condición, o los valores posibles de dicha condición, denominada condición disparadora.
Procedimiento	Dividir produce dos o más nuevos escenarios, uno por cada valor de la condición disparadora. Dicho valor suele formar parte del título del nuevo escenario para distinguirse, y dependiendo del tipo de condición formará parte de la precondición del nuevo escenario o de su objetivo. Los episodios que no contienen esta condición disparadora se trasladan tal cual a todos los nuevos escenarios. Por otro lado, los episodios con la condición disparadora se trasladan sólo al escenario correspondiente eliminando la condición. El escenario original se elimina. Deben revisarse los componentes actores, recursos y excepciones de los nuevos escenarios, pues debido al movimiento de episodios, puede dejar de estar presente un recurso o un actor, o excluirse alguna excepción de alguno de estos nuevos escenarios.
Propósito	Esta operación ayuda a evitar escenarios confusos, facilitando su legibilidad y reduciendo así la complejidad.

Fusionar	
Oportunidad	Esta operación puede aplicarse a escenarios que representan situaciones que ocurren una detrás de la otra sin una brecha temporal entre ellos, compartiendo la misma ubicación geográfica y temporal, y con objetivos acoplables. Una brecha temporal ocurre cuando existe un intervalo de tiempo largo entre dos escenarios.
Procedimiento	Fusionar copia los episodios de cada escenario original a un nuevo escenario en el orden correspondiente. El objetivo del nuevo escenario debe abarcar ambos escenarios, en general se construye a partir del objetivo del último escenario fusionado. Los actores, recursos y excepciones del nuevo escenario se obtienen uniendo los correspondientes componentes de los escenarios fusionados. La precondición se construye a partir de la precondición del primer escenario fusionado, agregándole precondiciones de carácter externo de los restantes escenarios fusionados. El contexto temporal se construye por generalización como suma de los contextos temporales originales. La ubicación geográfica debe ser la misma en todos los escenarios originales y, por ende, se traslada al nuevo escenario. Se eliminan los escenarios originales.
Propósito	Esta operación permite reducir la cantidad de escenarios y facilitar entonces su administración.

Separar	
Oportunidad	Esta operación puede aplicarse a un escenario si éste tiene una brecha temporal entre episodios o cuando se detecta un contexto temporal muy extenso. Un contexto temporal largo puede detectarse por el contexto en sí mismo o por la secuencia de los episodios. Esta operación también puede considerarse cuando se descubre más de una ubicación geográfica en un escenario. Es decir, el escenario está representando más de una situación contradiciendo la definición misma de escenario. La descomposición del escenario debería producir escenarios con un contexto temporal bien acotado.

Procedimiento	Separar utiliza la brecha temporal como guía para la descomposición del escenario en varios escenarios según la cantidad de demoras detectadas. Esta operación copia en un nuevo escenario todos los episodios que preceden a la brecha temporal y aquellos episodios que siguen a la brecha temporal se copian en otro nuevo escenario. El escenario original se elimina. Las precondiciones del segundo y siguientes nuevos escenarios reflejan el orden de precedencia establecido. Los componentes actores y recursos de cada nuevo escenario deben re-hacerse. Los objetivos de los nuevos escenarios serán más acotados que el objetivo original, restringiéndose al conjunto de episodios que contienen.
Propósito	Esta operación permite reducir la complejidad y comprender mejor las situaciones.

La Figura 6-6 muestra la operación Factorizar aplicada a dos escenarios con varios episodios comunes que tienen un objetivo específico; se crea un nuevo escenario conteniendo estos episodios; los escenarios originales se re-escriben, reemplazando los episodios comunes extraídos por el título del nuevo escenario. La Figura 6-7 ejemplifica la operación Consolidar aplicada a dos escenarios superpuestos, produciendo un único escenario que contiene los episodios comunes mientras los episodios no compartidos se incorporan con una sentencia condicional. Los escenarios graficados en ambas figuras corresponden al mismo caso de estudio de la Figura 6-3.

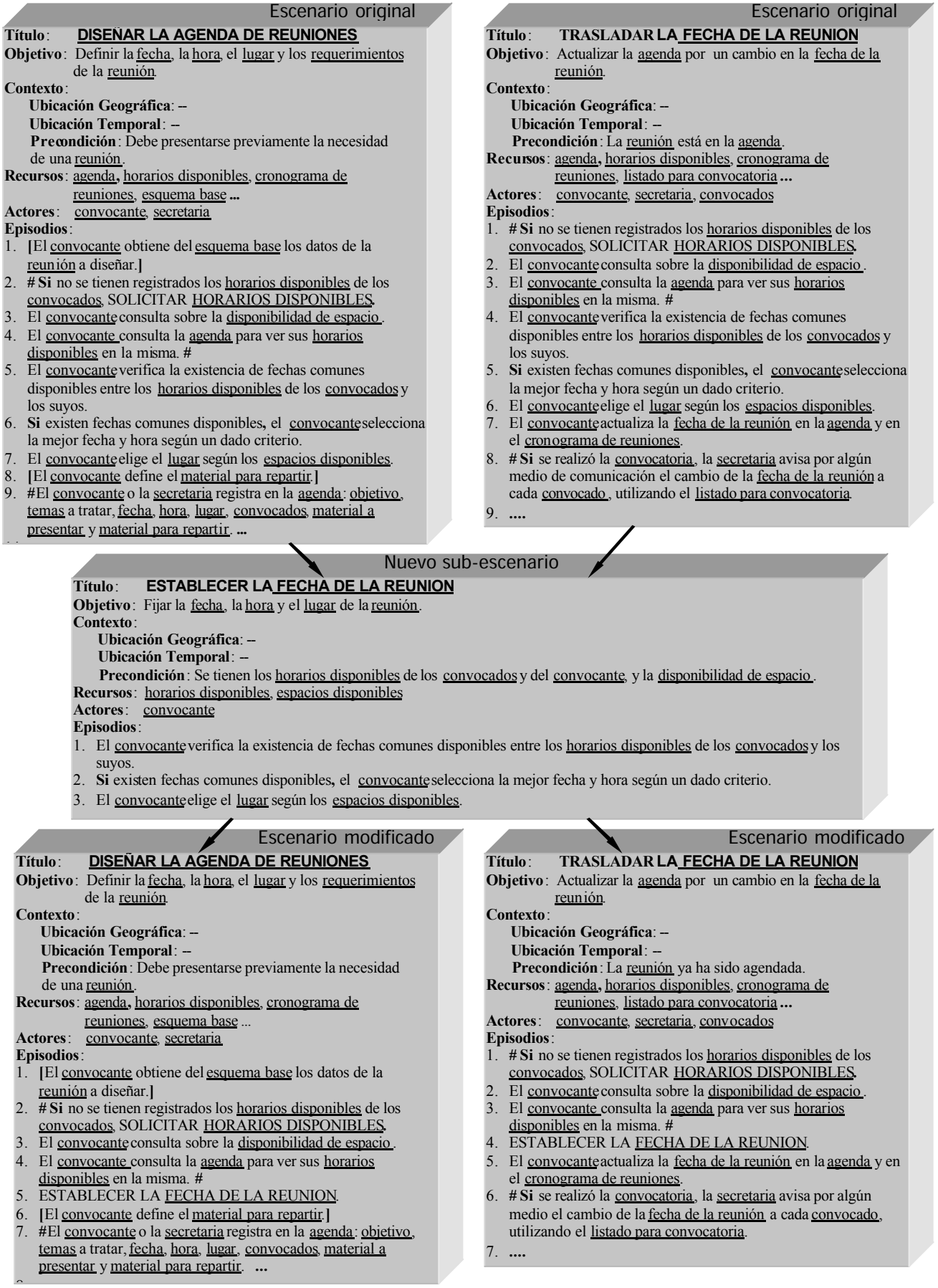


Figura 6-6. Ejemplo de la operación Factorizar

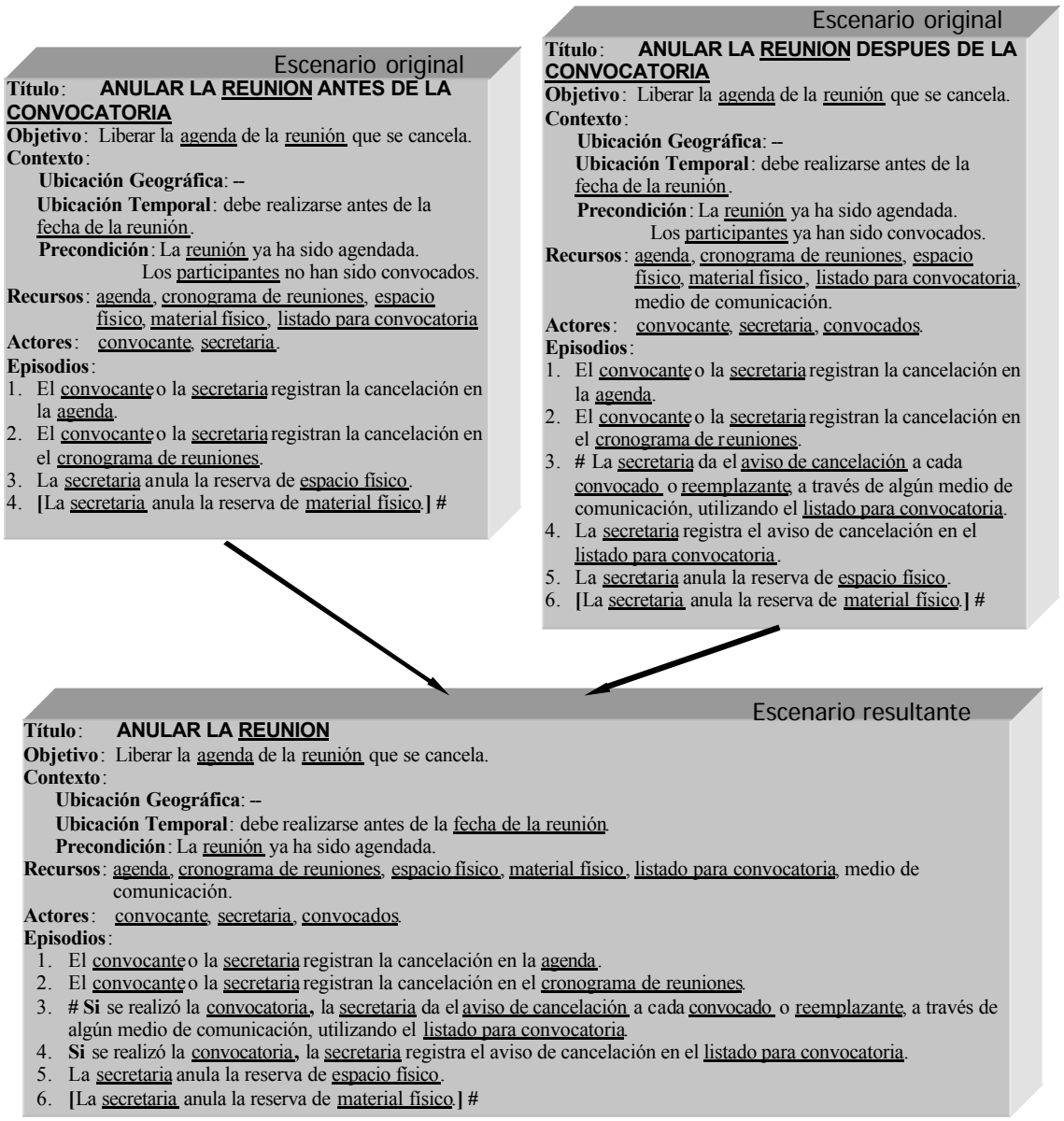


Figura 6-7. Ejemplo de la operación Consolidar

Cuando se aplican las operaciones de composición, el objetivo del escenario resultante deberá extenderse para abarcar la situación completa, especialmente al fusionar; en el caso de consolidar, el objetivo se extiende por generalización. Por el contrario, cuando se aplica descomposición, el objetivo del nuevo escenario puede ser menos global que el original.

Dado que Consolidar y Dividir son operaciones opuestas, después de consolidar, el escenario resultante puede ser considerado para una división, retornando al estado original y viceversa. Se requieren entonces criterios para decidir la aplicación de esta operación. Consolidar deberá aplicarse si aparecen pocos episodios condicionales en el escenario resultante, de lo contrario debe evitarse. Por otro lado, Dividir deberá elegirse sólo si varios episodios con la misma condición serán afectados por la operación.

Fusionar y Separar son operaciones opuestas también; por lo tanto, ambas requieren criterios de decisión sobre la mejor dirección a tomar. La clave es la brecha temporal, tanto respecto a escenarios como episodios. Una demora entre dos episodios puede considerarse como una brecha temporal sólo si no puede ser abarcada por el contexto temporal del escenario. La aplicación de estas dos operaciones es mucho más sencilla que en las otras pues es más evidente encontrar una brecha temporal o más de una ubicación geográfica.

Las operaciones relativas al aspecto de granularidad también requieren criterios de decisión para evitar ciclos infinitos pasando de un estado al opuesto. Aplanar deberá aplicarse sólo si los episodios una vez aplanados en el escenario resultante tienen el mismo nivel de detalle que el resto de los episodios y si efectivamente el sub-escenario original no representa él mismo una situación, lo que puede confirmarse por la falta de significado de su objetivo. Por el contrario, Factorizar deberá usarse sólo si se puede encontrar un objetivo significativo para el nuevo sub-escenario como consecuencia de haber descubierto una situación específica y real dentro de un escenario más abarcador.

(3.2) DEFINIR

Es necesario identificar diferentes relaciones entre los escenarios para luego integrarlos. A continuación se describen los tipos de relaciones que se pueden establecer:

Relación de Jerarquía
Es aquella que se establece entre un escenario y un sub-escenario. Esta relación ocurre naturalmente mientras se describen los escenarios o cuando se reorganizan. El comportamiento de un escenario se describe a través de un conjunto de episodios, los cuales pueden ser acciones simples o sub-escenarios; el último caso ocurre cuando la acción es más compleja o es común a varios escenarios. Un escenario puede contener más de un sub-escenario o ninguno. Un sub-escenario puede ser incluido en uno o más escenarios y él mismo puede contener sub-escenarios, permitiendo varios niveles de profundidad en la jerarquía. Entonces, se define una jerarquía como un conjunto compuesto por un escenario y sus sub-escenarios, siendo que el escenario raíz de la jerarquía no sea a su vez un sub-escenario.
Relación de Superposición
Es aquella que se establece entre escenarios con partes comunes. Esta relación se observa principalmente cuando varios episodios comunes están presentes en diferentes escenarios y, por lo tanto, comparten actores y aparecen probablemente recursos comunes.
Relación de Orden

Es aquella que se establece entre dos jerarquías cuando una precede a la otra. Una jerarquía que ocurre antes que otras determina una relación de orden temporal parcial entre ellas. Una jerarquía puede tener cero o más predecesores y cero o más sucesores. Una secuencia se establece cuando una jerarquía está inmediatamente precedida por otra. Dado que la segunda puede estar también precedida por una tercera, un gran número de jerarquías pueden estar involucradas en una secuencia, la cual comienza por lo menos con una jerarquía cabecera. Una secuencia puede tener más de una jerarquía cabecera y también más de una jerarquía cola.

Relación de Excepción

Es aquella que se establece entre un escenario y los escenarios que tratan sus excepciones. Cuando un escenario tiene una excepción, se describe la causa de la misma y puede especificarse un tratamiento de la excepción a través de un escenario. Un escenario puede hacer referencia a uno o más escenarios excepción. Un mismo escenario excepción puede tratar excepciones ocurridas en diferentes escenarios.

(3.3) INTEGRAR

Esta actividad consta de cinco pasos, e implementa en sí misma el enfoque middle-out. Sus tres primeros pasos son bottom-up y los dos últimos son top-down. Los primeros cuatro pasos manejan principalmente información sintáctica, pero en el último se requiere más un conocimiento semántico. En esta actividad, primero, se agrupan los escenarios en jerarquías y luego éstas se agrupan en secuencias. Finalmente, estas secuencias son usadas para construir los escenarios integradores.

La Figura 6-8 muestra dos ejemplos de escenarios integradores extraídos de distintos casos. Cabe destacar que los escenarios integradores se producen sólo para organizar el conjunto disperso de escenarios, y no usan los componentes Actores y Recursos, dado que en general representarían la lista completa de actores y recursos participantes en todo el conjunto de escenarios.

Sistema de Agenda de Reuniones

Título: **ADMINISTRAR LA AGENDA DE REUNIONES**
Objetivo: Administrar la agenda de reuniones eficiente y efectivamente.
Contexto:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Existe un tema a tratar o resolver por dos o más personas.
Recursos: --
Actores: --
Episodios:
1. REQUERIR UNA REUNIÓN.
2. DISEÑAR LA AGENDA DE REUNIONES.
3. ORGANIZAR LA REUNIÓN.
4. **Si** la reunión no se canceló **entonces** ASISTIR A LA REUNIÓN.

Excepciones:

Sistema Nacional para la obtención de Pasaportes

Título: SOLICITAR UN NUEVO PASAPORTE.
Objetivo: Entregar un pasaporte a un solicitante por primera vez.
Contexto:
Ubicación geográfica: División Documentos y Certificados, División Índice General, División Dactiloscopía, División Legajos y Antecedentes.
Ubicación geográfica:
Precondición: El solicitante nunca ha pedido un pasaporte.
Actores: --
Recursos: --
Episodios:
1. LLENADO DE FORMULARIO.
2. CONTROL DE DOCUMENTACIÓN.
3. #SACAR FOTOGRAFÍA.
4. PAGAR TRÁMITE.
5. **Si** el solicitante tiene un número de identificación, **Entonces** OBTENCIÓN DE HUELLAS DIGITALES PARA CONTROL.
6. **Si** el solicitante no tiene un número de identificación, **Entonces** OBTENCIÓN DE HUELLAS DIGITALES PARA LEGAJO. #
7. DERIVACIÓN A CABINA DE RECEPCIÓN.
8. ARMADO DE PASAPORTE ORIGINAL.
9. CONTROL DE PRONTUARIO.
10. CONTROL DACTILOSCÓPICO.
11. VERIFICACIÓN FINAL Y CERTIFICACIÓN DE PASAPORTE.
12. ENTREGA DE PASAPORTE.
13. ENVÍO DEL LEGAJO DEL SOLICITANTE AL ARCHIVO GENERAL DE LEGAJOS.

Excepciones:
El solicitante no retira el pasaporte en el período previsto (ENVIO A INCINERACIÓN DE PASAPORTES NO RETIRADOS).
Un pasaporte es observado (DERIVACIÓN DE PASAPORTES OBSERVADOS).

Figura 6-8. Ejemplos de Escenarios Integradores

La integración utiliza esencialmente las relaciones de jerarquía, de orden y de excepción. La relación de superposición puede usarse en esta actividad, permitiendo una integración por generalización como un paso más en el proceso general de integración. Es aplicable cuando se cuenta con un conjunto de escenarios a los que se les aplicó la operación División o naturalmente al Describir se obtuvieron escenarios superpuestos con un objetivo común, es decir, se está frente a un conjunto de escenarios que representan situaciones con variantes. Pero en la experiencia ha sido descartada, por su bajo aporte en mejorar la comprensión de los escenarios.

(3.3.1) CONSTRUIR JERARQUÍAS DE ESCENARIOS

Como ya se indicó en ***(3)***, la actividad de integración aspira a proveer una visión global del UdeD atenuando la importancia y la presencia de detalles. En el conjunto de escenarios a integrar, los sub-escenarios y los escenarios que atienden una excepción deben ser opacados. Es así que se destacan los escenarios cabecera constituyendo jerarquías de escenarios donde aquellos otros son absorbidos por los escenarios cabeceras de jerarquías.

Una jerarquía es un conjunto de escenarios conectados por una relación jerárquica. Toda jerarquía tiene un escenario raíz o cabecera, el cual no debe estar mencionado como sub-escenario ni escenario excepción por ningún otro escenario. Una jerarquía se identifica por su escenario raíz.

Un escenario aislado, aquél que no pertenece a ninguna jerarquía, puede ser en sí mismo una jerarquía o puede corresponder al tratamiento de una excepción de algún escenario de mayor nivel aún no identificado. Debe evaluarse a cuál de los dos casos incumbe. Si se trata de un escenario excepción, apartarlo temporalmente para luego adosarlo en el paso ***(3.3.5)*** al componente Excepciones del escenario integrador que le corresponda.

Adicionalmente, se debe determinar para cada sub-escenario si tiene sentido que ocurra aisladamente de otra situación, es decir, que el sub-escenario tenga identidad propia como una situación sin depender de otra más abarcadora para su realización. En caso afirmativo, este sub-escenario debe considerarse también como una jerarquía.

Para el siguiente paso se requiere construir ciertos componentes de las jerarquías: título, objetivo, precondiciones, recursos y restricciones. En el caso de título y objetivo, éstos son directamente aquellos del escenario raíz, pero los restantes componentes deben construirse a partir de la jerarquía completa, es decir, incluyendo información de los sub-escenarios.

Las precondiciones, recursos y restricciones de la jerarquía no se obtienen del escenario raíz sino que son parte de la jerarquía completa. Para establecer las precondiciones de una jerarquía, primero se deben unir las precondiciones de todos los escenarios de la jerarquía, y luego se eliminan de la unión las precondiciones requeridas por un sub-escenario pero satisfechas por otro sub-escenario de la jerarquía. Las restricciones de la jerarquía se obtienen de la misma forma. Los recursos de la jerarquía se determinan como la unión de todos los recursos de los escenarios de la jerarquía, eliminando aquellos recursos que son a su vez generados por algún sub-escenario. Los episodios de la jerarquía se obtienen por aplanamiento de los episodios de los sub-escenarios.

(3.3.2) DETECTAR ORDEN PARCIAL ENTRE JERARQUÍAS

Esta actividad se basa en la comparación entre las precondiciones, recursos o restricciones de una jerarquía, y el título, objetivo o episodios de otra

jerarquía. Si una precondición de una jerarquía identifica un estado inicial que es satisfecho por otra jerarquía, entonces existe un orden parcial entre ambas jerarquías. Lo mismo puede ocurrir con un recurso requerido en una jerarquía y que es producido por otra jerarquía. Además, las restricciones de los episodios pueden ser satisfechas por otra jerarquía, por lo tanto, se establece un orden parcial. Estas restricciones pueden usarse en la comparación cuando se refieren a estados de los recursos o ciertas condiciones que deben resolverse previamente.

La comparación entre dos jerarquías termina cuando se detecta un orden parcial o la imposibilidad de lograrlo. Si el orden parcial entre dos jerarquías se establece sólo bajo cierta condición, la relación se etiqueta con esta condición. Para reducir la búsqueda de órdenes parciales, es mejor chequear primero contra el título y el objetivo, y luego contra los episodios de ser necesario. Si una comparación es exitosa, un orden parcial es encontrado, y se pasa a la comparación con las restantes jerarquías. Si ninguna comparación fue exitosa, entonces no hay orden parcial entre ambas jerarquías, continuando con siguientes jerarquías. Las comparaciones usadas en este paso son en el siguiente orden:

1. Precondición de una jerarquía contra el título de la otra jerarquía.
2. Recursos de una jerarquía contra el título de la otra jerarquía.
3. Restricciones de una jerarquía contra el título de la otra jerarquía.
4. Precondición de una jerarquía contra el objetivo de la otra jerarquía.
5. Recursos de una jerarquía contra el objetivo de la otra jerarquía.
6. Restricciones de una jerarquía contra el objetivo de la otra jerarquía.
7. Precondición de una jerarquía contra los episodios de la otra jerarquía.
8. Recursos de una jerarquía contra los episodios de la otra jerarquía.
9. Restricciones de una jerarquía contra los episodios de la otra jerarquía.

Cuando el orden parcial se establece utilizando los episodios en la comparación, se debe evaluar si la precedencia corresponde a todo el conjunto de episodios de la jerarquía o sólo a partir del episodio donde se satisfizo la comparación. En este último caso, considerar si no debiera aplicarse la operación Separar sobre el escenario al que pertenece este episodio, y luego re-armar la jerarquía, y re-evaluar el orden parcial.

(3.3.3) CONSTRUIR SECUENCIAS DE JERARQUÍAS

Esta actividad comienza con la revisión de las relaciones de orden parcial detectadas entre las jerarquías, con el fin de clarificar cuál es el conjunto de jerarquías que inmediatamente preceden a otra. Primero, se

eliminan todos los órdenes parciales derivados por transitividad. Se dice que existe transitividad entre las jerarquías A, B y C, si A y B preceden a C pero además A precede a B, ver la Figura 6-9.

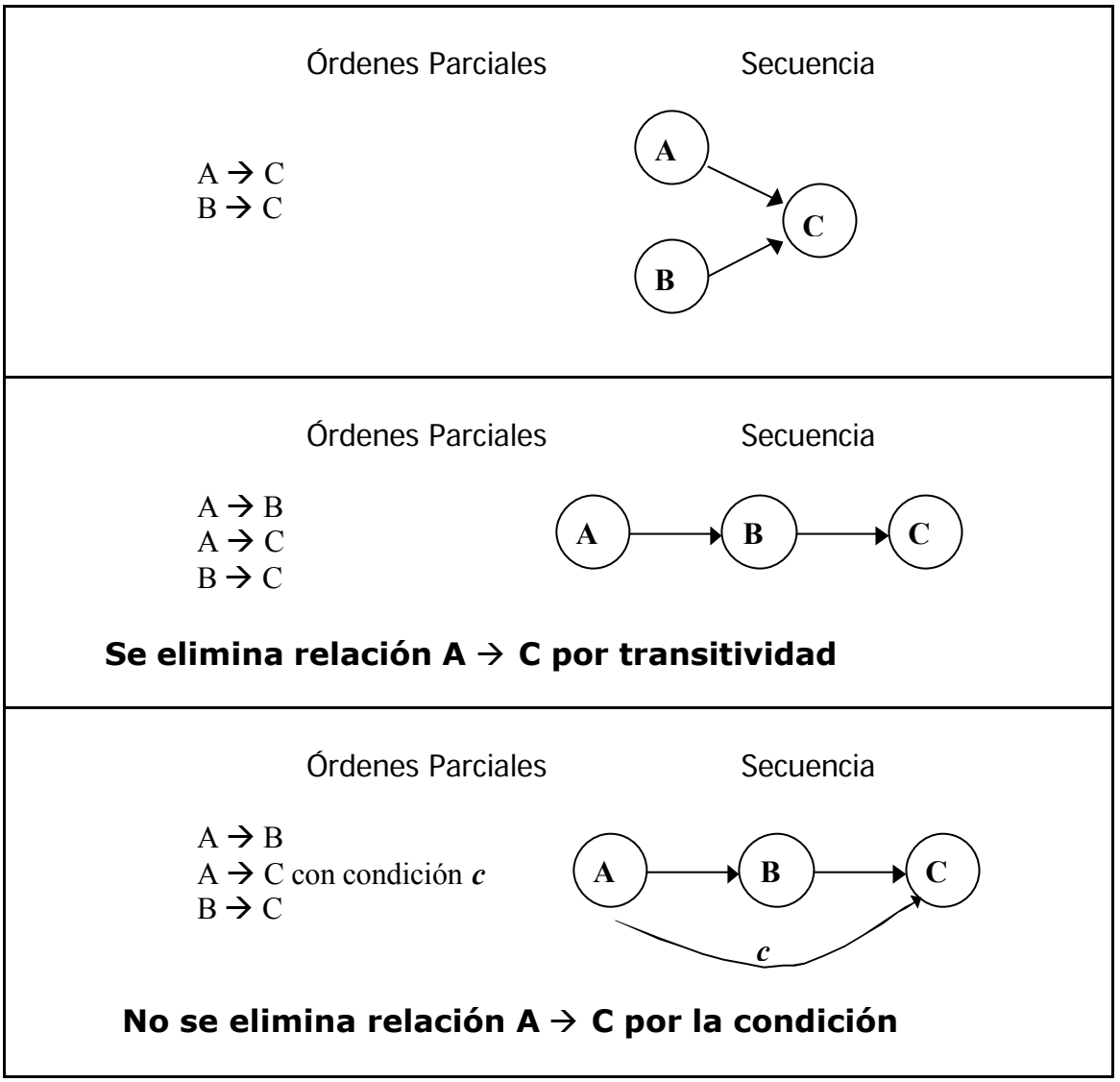


Figura 6-9. Ejemplos de Construcción de Secuencias

Las transitividades no son fáciles de eliminar. Sin embargo, éstas pueden eliminarse agrupando todas las relaciones de orden parcial con la misma jerarquía a la izquierda de la relación. Cualquier orden parcial adicional entre las jerarquías sobre la parte derecha de la relación se usa para identificar una transitividad. Nueva información proveniente del UdeD puede ser elicitada cuando se detectan brechas temporales entre las jerarquías. Esto sucede cuando dos jerarquías en secuencia no coinciden apropiadamente, debido a que, a una o a ambas jerarquías les faltan algunas acciones que realmente se ejecutan en el UdeD. Las jerarquías aisladas son secuencias en sí mismas.

(3.3.4) CONSTRUIR ESQUELETOS INTEGRADORES

En este paso sólo se construye el componente Episodios de los escenarios integradores. Primero se construye un escenario principal (de más alto nivel) y luego se pueden obtener niveles intermedios de escenarios integradores.

Se le asigna un identificador a cada secuencia obtenida en el paso anterior. Cada secuencia será un episodio del escenario integrador principal, pero no tiene establecida un orden de precedencia respecto a las restantes. Luego, todas las secuencias, a través de su identificador, se incorporan al componente Episodios del escenario integrador principal marcando con indicadores de no-secuencia (#) al principio y al fin del conjunto de episodios.

Si una secuencia representa una jerarquía sola, entonces el identificador de la secuencia se reemplaza por el título del escenario raíz de la jerarquía. Si una secuencia está compuesta por más de una jerarquía, se crea un escenario integrador de nivel intermedio, adosando el identificador al título y las jerarquías de la secuencia pasan a formar parte del componente Episodios del escenario integrador intermedio, considerando sus precedencias o no secuencias. Esto se aplica cada vez que se detectan sub-secuencias dentro de grupos no secuenciales.

(3.3.5) COMPLETAR COMPONENTES DE ESCENARIOS INTEGRADORES

En este paso es frecuente agregar nueva información semántica. Se miran los episodios de los escenarios integradores para completar los restantes componentes, comenzando por los de menor nivel, para ir propagando sus títulos a los nombres usados en los episodios. Este paso actúa como una clase de verificación final.

El título y el objetivo deben escribirse preservando cohesión, es decir, sin usar conjunciones en las acciones. Se escribe el contexto en función de los episodios involucrados; en principio la precondición correspondería a la precondición del primer episodio. Las descripciones deben ser escritas utilizando los símbolos del LEL.

Volver a cuestionarse qué podría fallar en cada episodio que ahora es un escenario, para detectar causas de excepciones. Si se habían apartado escenarios excepción, vincularlos al escenario integrador correspondiente, tener en cuenta su precondición, podría ser la causa de la excepción.

También buscar en los episodios que son escenarios, si éstos a su vez tienen escenarios que atienden excepciones pero que no cumplen con el objetivo del escenario episodio. Si esto ocurre entonces en los episodios secuenciales siguientes del escenario integrador todos los episodios deben ser condicionales, siendo la condición la negación de la causa de la excepción encontrada.

Si alguna Ficha de Información Anticipada quedó con información no utilizada, revisarla para determinar si dicha información puede volcarse en algún escenario integrador, como por ejemplo algún RNF con un ámbito más amplio que una situación real ya descripta.

(4) VERIFICAR

Esta actividad se realiza por lo menos dos veces durante el proceso de construcción de escenarios, la primera vez sobre el conjunto de escenarios completamente descriptos y la segunda después de la actividad Organizar. Se realiza siguiendo una guía con heurísticas de verificación. Como consecuencia de esta actividad, se producen dos listas de DEOs: una para modificar los escenarios en la actividad Describir, y la otra para actualizar el LEL. La verificación se divide en Verificación Intra-Escenarios, Verificación Inter-Escenarios y Verificaciones contra el LEL.

Es importante enfatizar que el proceso de verificación está dirigido por controles sintácticos, referencias cruzadas con el LEL, el modelo de Escenarios y por una guía de control. A continuación se detallan algunas heurísticas de la guía.

Verificación Intra-Escenario	
Verificar la sintaxis	Controlar la completitud de cada componente.
	Controlar la existencia de más de un episodio por escenario.
	Controlar la sintaxis de cada componente según se establece en el modelo de Escenario.
Verificar la relación entre componentes	Controlar que todo actor participe en al menos un episodio.
	Controlar que todo actor mencionado en loa episodios sea incluido en el componente actores.
	Controlar que todo recurso sea usado en al menos un episodio.
	Controlar que todo recurso mencionado en los episodios sea incluido en el componente recursos.
Verificar la semántica	Controlar la coherencia entre el título y el objetivo.
	Asegurar que el conjunto de episodios satisfaga el objetivo y esté dentro del contexto.
	Asegurar que las acciones presentes en las precondiciones ya han sido satisfechas.
	Asegurar que los episodios contengan solamente acciones a ser realizadas.

	Asegurar el mismo nivel de detalle en la descripción del escenario.
Verificación Inter-Escenario	
Verificar la existencia de sub-escenarios	Controlar que cada episodio identificado como un sub-escenario exista dentro del conjunto de escenarios.
	Controlar que episodios del sub-escenario no estén incluidos en el componente Episodios del escenario.
	Controlar que toda excepción sea tratada por un escenario que existe o mediante alguna acción simple.
Verificar el contexto	Controlar que toda precondición sea un hecho incontrolable (externo) o satisfecho por otro escenario.
	Controlar la coherencia entre las precondiciones de los sub-escenarios y las de los escenarios correspondientes.
	Controlar que la ubicación geográfica y temporal de los sub-escenarios sea coherente con la de los escenarios que los contienen.
Verificar la equivalencia en el conjunto de escenarios	Controlar que la coincidencia de objetivos tenga lugar solo en situaciones diferentes.
	Controlar que la coincidencia de episodios tenga lugar solo en situaciones diferentes.
	Controlar que la coincidencia de contextos tenga lugar solo en situaciones diferentes.
Verificación contra el LEL	
Controlar que todo símbolo del LEL es identificado al mencionarse en un escenario.	
Controlar el uso correcto de símbolos del LEL.	
Controlar que los actores sean preferentemente símbolos del tipo Sujeto.	
Controlar que los recursos sean preferentemente símbolos del tipo Objeto.	
Controlar que los escenarios cubran todos los impactos de los símbolos Sujeto del LEL.	

Usando las listas de verificación de DEOs, se modifican los escenarios y el LEL. Si se necesitan realizar correcciones mayores, entonces puede requerirse una nueva verificación. Cuando el ciclo Describir-Verificar termina, la actividad Organizar puede proveer un conjunto más grande de escenarios, los cuales a su vez deberían ser verificados, comenzando un posible ciclo Describir-Organizar-Verificar.

Aunque la verificación de los EA puede realizarse siguiendo variadas técnicas, en esta tesis se recomienda un proceso de inspección, que se detalla en la sección 6.2, basado en la propuesta original de Fagan [Fagan 76], pero

utilizando una técnica de lectura durante la preparación adaptada de la propuesta de Porter [Porter 95] y ajustada al modelo de Escenario.

(5) VALIDAR

Los escenarios son validados con los usuarios, generalmente realizando entrevistas estructuradas o reuniones. Durante la validación, debe ser considerado con especial atención el componente Dudas (que puede ser agregado durante la actividad Describir y que se elimina durante la presente actividad) de aquellos escenarios que lo presenten.

Es importante mencionar que a pesar que se utilice una descripción estructurada, los escenarios son escritos en LN, empleando el vocabulario propio de los usuarios y describiendo una situación específica bien limitada en la que ellos están involucrados.

La actividad de validación se realiza para detectar DEOs en los escenarios, pero al igual que en la verificación también es posible detectar pero con mucha menor frecuencia algún defecto en el LEL, es por ello que podrán producirse a partir de esta actividad dos listas de DEOs. Los errores son detectados principalmente leyendo cada escenario al usuario, algunas omisiones aparecen durante la lectura pero otros preguntando sobre la falta de información o de detalles en el escenario. Las discrepancias pueden aparecer durante las reuniones cuando existen diferencias entre los usuarios o cuando se analiza la información obtenida. Los problemas se registran mediante una lista de DEOs.

El principal objetivo de la validación es confirmar la información elicitada y detectar DEOs; sin embargo, un efecto colateral de este proceso es elicitar nueva información. La validación de un conjunto de escenarios, después del proceso de verificación, permite confirmar que las situaciones del UdeD acontecen, y que han sido registradas con la percepción de los actores del UdeD a cargo de la lectura y discusión de los escenarios.

Existen otras posibilidades para validar escenarios, como: ejercicios de role playing con los participantes, ejemplificando con situaciones concretas, animación por medio de representaciones pictóricas o simulando el comportamiento de los episodios mediante ambientes orientados a máquinas de estado. La validación de escenarios está abierta a la imaginación y las habilidades técnicas de simular descripciones de escenarios. Este es un problema abierto en el que podrían surgir nuevas contribuciones que mejoren la actividad.

6.1.3. Observaciones

Existe una gran variedad en las formas de relacionar escenarios. Varios autores han propuesto relaciones tales como las asociaciones de uso y de extensión de Jacobson [Jacobson 94b], las conexiones semánticas de Booch

[Booch 94], los vínculos temporales de Dano [Dano 97], las relaciones de superposición, equivalencia y subconjunto de Breitman y Leite [Breitman 98], las relaciones implícitas de Cockburn [Cockburn 00] según el nivel de detalle en resumen, usuario y sub-función, entre otras. Estas relaciones fueron identificadas y buscadas con otros propósitos, las relaciones que se detectan en la SDRES son con el fin de integrar escenarios y por ende difieren de las utilizadas por otros autores.

De Carroll [Carroll 95] se obtuvo una mejor comprensión de los aspectos cognitivos del desarrollo basado en escenarios y también una confirmación que los escenarios deben comenzar en el UdeD y no solamente de la interfaz del software. Esta idea fue asimismo reafirmada por Zorman [Zorman 95], quien definió escenarios como situaciones; además se utilizó su estudio de representación de escenarios. Potts [Potts 94] mostró la relevancia de relacionar escenarios con objetivos y, por lo tanto, esto probó la importancia del conocimiento previo del meta modelo orientado a objetivos propuesto por Dardenne et al. [Dardenne 93]. En particular, esta tesis difiere de Potts [Potts 94] porque en ella se usa una jerarquía de escenarios en lugar de una jerarquía de objetivos y la construcción de jerarquías en SDRES usa una estrategia bottom-up contra la descomposición de objetivos de Potts.

Dano et al. [Dano 97] consideran que los casos de uso y los escenarios son intercambiables porque ellos capturan formas específicas de usar la funcionalidad del sistema, mientras que en esta tesis los escenarios capturan situaciones del dominio de la aplicación. Por otro lado, [Dano 97] propone la utilización de formalismos como una alternativa al LN. En esta tesis se usa el LN con reglas para elicitar y luego modelar escenarios.

Esta tesis comparte con Sutcliffe [Sutcliffe 97b] la conveniencia de integrar métodos para la producción de requisitos. Por otro lado, la idea de desarrollar un prototipo que actúe como un "demostrador de conceptos", como propone este autor, solo ayuda a establecer requisitos concernientes a la interfaz hombre-máquina y que son específicos al artefacto a construir, dejando fuera otros aspectos del dominio de la aplicación.

Las operaciones presentadas en la actividad Organizar (ver sub-sección 6.1.2) se inspiraron en el trabajo de Breitman [Breitman 98] que descubrió, del estudio de conjuntos de escenarios, que sus autores usaban cierto tipo de operaciones en la producción de escenarios. La diferencia en esta tesis es que las operaciones se definieron con el objetivo de producir un conjunto organizado de escenarios. Las operaciones aplicadas por Breitman también basadas en relaciones tienen por objetivo la evolución de los escenarios de especificación a escenarios de diseño y luego a escenarios de implementación. En esta tesis, los escenarios futuros parecieran corresponderse, debido a su nombre, con los escenarios de especificación, aunque en [Breitman 98] se menciona que los escenarios de especificación reflejan el macrosistema tal cual es mientras los escenarios de diseño muestran el macrosistema según la aplicación. Es decir, los escenarios de especificación reflejan lo actual y los escenarios de diseño reflejan el software. Los escenarios futuros presentados en esta tesis y los escenarios de diseño de Breitman miran mundos muy

parecidos, pero desde puntos de vista muy diferentes. Los escenarios futuros miran el futuro desde el negocio y los de diseño desde el software. Además, los escenarios de diseño se deben hacer al mismo tiempo que el diseño mientras que los escenarios futuros son muy anteriores.

Se debe notar un grado de paralelismo alto entre las operaciones presentadas por [Breitman 98] y las presentadas en esta tesis en la sub-sección anterior. Estas similitudes tienden a esconder un aspecto relevante: las operaciones descriptas por [Breitman 98] tienen su eje central en que son semánticamente invariantes, ya que aspiran a que el software a desarrollar sea un fiel reflejo de lo ya modelado. En cambio las operaciones presentadas en esta tesis introducen modificaciones semánticas en el conjunto de EA para describir mejor el UdeD, y en los EF para describir mejor el rol que tendrá el sistema en el proceso del negocio.

Características	Escenario General	Escenario Integrador
Enfoque	TOP-DOWN	MIDDLE-OUT
Punto de construcción	AL COMIENZO	CERCA DE FINALIZAR
Requiere refinamiento	SÍ	NO
Esfuerzo de consistencia	ALTO	MUY BAJO
Vista completa del macrosistema	APROXIMADA	PROXIMA
Vista entre escenarios	POBRE	CLARA
Permite dividir el problema	SÍ	NO
Permite chequear consistencia	SÓLO SI SE REFINA	SÍ

Tabla 6-2. Cuadro comparativo de construcción de escenarios

De la experiencia extraída a lo largo de estos años usando escenarios, se puede observar que existen dos formas de organizarlos a través del empleo del modelo de escenario, una denominada "escenarios generales" [Hadad 97] y la otra "escenarios integradores" [Leite 00]. Los escenarios generales son escenarios raíz o principales que se utilizan al seguir una estrategia top-down. Esta idea es similar a los escenarios primarios de Booch [Booch 94], a los escenarios fundamentales de Sutcliffe [Sutcliffe 97b] y a los casos de uso nivel resumen de Cockburn [Cockburn 00]. Es decir, el escenario general permite tener una visión abstracta del dominio de la aplicación desde el comienzo del proceso de construcción de los escenarios. Por otro lado, cuando los escenarios se desarrollan siguiendo las heurísticas descriptas en la sub-sección previa, no hay necesidad de tener un panorama general al comienzo del proceso. La Tabla 6-2 sintetiza las características de usar escenarios generales frente a escenarios integradores.

En una estrategia top-down, los escenarios generales se describen al comienzo del proceso de construcción de escenarios, a partir de un conocimiento pobre y muy general del dominio de la aplicación. Por el contrario

los escenarios integradores en un enfoque middle-out se describen como el último paso del proceso, después de haber adquirido un gran y detallado conocimiento del dominio. Lo mismo ocurre en un enfoque bottom-up.

Los escenarios generales necesitan un mantenimiento continuo a lo largo del proceso de construcción de manera de lograr consistencia con el resto de los escenarios, de otra manera, pierden su utilidad y deben separarse del conjunto. Como los escenarios integradores se construyen al final del proceso, son naturalmente consistentes con el resto de los escenarios.

Los escenarios generales dan una imagen aproximada del dominio de la aplicación al comienzo del proceso y con un gran esfuerzo de mantenimiento pueden proveer una visión similar a la de los escenarios integradores. Este esfuerzo está directamente relacionado con la posibilidad de verificar la consistencia entre escenarios. Los episodios de los escenarios integradores muestran las relaciones entre los escenarios, mientras que los escenarios generales muestran actividades que pueden o no ser escenarios.

En la práctica [Weidenhaupt 98], los escenarios generales se usan para dividir tareas, aunque éstas aún no son bien conocidas, entre el equipo de ingenieros. Esto no es posible ni deseable con el proceso presentado. Como observó Jackson (ver sub-sección 2.8.2), un enfoque top-down es útil para organizar un problema cuando éste es conocido previamente, pero no antes.

En la Tabla 6-3 se muestran los modos de construcción de escenarios basados en los enfoques top-down, bottom-up y middle-out, así como las fortalezas y debilidades que presenta cada enfoque. Cabe mencionar que en [Regnell 96b] se presentan dos enfoques para construir casos de uso: top-down y bottom-up. En el enfoque bottom-up, a diferencia de lo expresado en la Tabla 6-3, comienzan describiendo ejemplos de escenarios, identifican episodios comunes, generalizan los escenarios y sintetizan en casos de uso que los abarcan, mientras que en la modalidad top-down comienzan identificando actores y casos de uso, siguiendo una construcción similar a la expresada en la Tabla 6-3. Los autores sugieren el uso de ambos enfoques en forma iterativa en un mismo método.

En base a los casos desarrollados con escenarios, hay una ventaja interesante usando escenarios integradores mediante un enfoque middle-out, en lugar de una organización top-down. En esta tesis se establece que una estrategia middle-out produce un conjunto de escenarios que, aunque representa distintas situaciones del UdeD, está organizado siguiendo los principios de modelado de la ingeniería de software. Aunque la estrategia de construcción está basada en un modelo de representación (Figura 5-3), los aspectos de composición / descomposición pueden ser de interés general para la construcción de escenarios.

Características	ENFOQUES		
	TOP-DOWN	**BOTTOM-UP**	**MIDDLE-OUT**
Punto de Arranque	Encontrar grandes funcionalidades. Escribir EG.	Buscar acciones atómicas. Escribir episodios.	Encontrar situaciones derivando del LEL. Escribir escenarios.
Etapa de Construcción	Buscar escenarios detallados que satisfagan los EG.	Agrupar episodios en escenarios.	Completar escenarios. Elicitar nuevos a partir de éstos.
Punto de Finalización	Actualizar los EG en base a los escenarios descriptos.	Reorganizar los escenarios. Construir los EI.	Reorganizar los escenarios. Construir los EI.
Vista global del macrosistema	Desde el comienzo con los EG. En general, es una vista inicial poco real.	Al final con los EI. Es una vista más próxima a la real.	Al final con los EI. Es una vista más próxima a la real.
Fortalezas	Los EG ayudan a definir el alcance del problema. Permite la división del problema en funciones de alto nivel. Es útil cuando se conoce el negocio.	Captura al inicio funciones elementales del problema.	Brinda un punto de arranque para los ingenieros. Facilita la visión de las relaciones entre los escenarios. Es útil cuando se tiene un bajo o mediano conocimiento del UdeD.
Debilidades	Se divide el problema partiendo de un conocimiento escaso y general del mismo. No permite una fácil visualización de las relaciones entre escenarios. Requiere el mantenimiento de los EG.	No se tiene una idea general del problema al inicio. Gran esfuerzo en encontrar detalles sin una guía. Dificultad en comprender los detalles sin una vista global. Dificultad para delimitar los escenarios.	No se tiene una idea general del problema al inicio (aunque se la logra progresivamente). Existe la necesidad de la construcción de los EI.

Tabla 6-3. Enfoques para construir escenarios

Un trabajo previo en integración de escenarios [Glinz 95] usaba un punto de vista de interacción con el usuario y se enfocaba en el aspecto de comportamiento de escenarios. Glinz usaba gráficos de estado como medio para modelar escenarios, explorando no sólo la capacidad de descomposición de los gráficos de estado sino también los aspectos de concurrencia. Por lo tanto, esto permitía no sólo la integración de varios escenarios en uno, sino también el análisis de los escenarios debido a problemas de consistencia. Contrariamente a Glinz, el enfoque de integración de escenarios presentado en este capítulo trata con aspectos de comportamiento de los escenarios (episodios), pero también con otra información de contexto, tal como actores, recursos, objetivo y precondiciones. Por otro lado, la propuesta de esta tesis está fuertemente basada en heurísticas.

Desharnais et al. [Desharnais 98] describen un enfoque de escenarios basado en estados relacionales y lo aplica a escenarios secuenciales. Los escenarios aquí son descripciones atómicas, en el estilo Z, de interacciones entre el sistema y el ambiente representado por una tripla (T, Re, Rs), siento T un espacio y, Re y Rs relaciones disjuntas sobre T. El enfoque asume que los escenarios satisfacen ciertas descripciones de consistencia. A partir de los escenarios atómicos, se deriva una especificación funcional. Estos escenarios atómicos parecen de un nivel aún menor que los episodios de la Figura 5-3, y no consideran aspectos no funcionales.

Uno de los primeros artículos que trató con la organización de escenarios provino de la comunidad orientada a objetos [Cockburn 95]. En dicho artículo Cockburn presenta el problema de la explosión de escenarios y propone una jerarquía de objetivos como manera de resolverlo. Regnell [Regnell 96] también trata con el problema en una forma orientada a objetivos, su propuesta organiza los casos de uso en un modo top-down (nivel de ambiente, nivel de estructura, nivel de evento), pero además observa que para comprender mejor un caso de uso debe haber una fase de síntesis donde los casos de uso son integrados en escenarios abstractos de uso.

El enfoque de CREWS-L'Ecritoire [Rolland 98] [Ben Achour 98] usa una combinación de objetivos y escenarios no sólo para elicitar sino también para organizar escenarios. A partir de la elicitación de objetivos, mediante el descubrimiento de casos, una estrategia dirigida por una plantilla conduce a los escenarios. Una vez que los escenarios están disponibles, se organizan en una forma jerárquica, también basados en el modelo de objetivos (relaciones AND/OR). Proveen algunas heurísticas para la gestión de calidad [Ben Achour 98]. Comparando este trabajo con el presentado en la tesis, se puede decir que en la tesis se usa directamente situaciones, en un modo bottom-up, para comprender el problema, y cuando se organizan los escenarios, éstos se integran con el mecanismo expuesto que encapsula la noción de relaciones AND/OR.

Sutcliffe et al. [Sutcliffe 98] presentan un método para IR basado en escenarios. Su representación básica es similar a la presentada en la tesis, con la diferencia que aquí se presentan jerarquías de escenarios / sub-escenarios (*n* niveles de estructuración) y ellos tienen un solo nivel de estructuración, es

decir, cada caso de uso puede tener *m* escenarios. Además presentan un método como un diagrama DFD [Sutcliffe 98, p.1074, Figura 1] donde se presentan cuatro pasos: elicitar casos de uso, analizar problemas genéricos, generar escenarios y validar escenarios. En este capítulo se ha mostrado un proceso más detallado, enfocando en la elicitación de escenarios a partir de la connotación del vocabulario de la aplicación.

El proceso de construcción presentado responde la pregunta cómo organizar escenarios cuando se tienen muchos de ellos. También responde a una pregunta habitual: ¿por dónde empezar? Esto se realiza mediante la elicitación de escenarios derivados del LEL. El proceso tiene una clara distinción entre crear los escenarios desde las diversas situaciones y luego organizarlos como un modelo, en el sentido de la ingeniería de software. Debe destacarse que el proceso presentado se basa en experiencias reales con el uso de escenarios, llevados a cabo desde 1996 a la fecha.

6.2. Inspección de escenarios

6.2.1. Características de la inspección

En este proceso de construcción de escenarios, se ha elegido la inspección como la forma de asegurar la calidad de los escenarios. Esta idea de aplicar inspecciones para mejorar la calidad de los escenarios es intuitiva y ya ha sido aplicada antes [Gough 95]. En el reporte industrial de CREWS [Weidenhaupt 98], se informó que 9 de 15 proyectos habían usado algún tipo de proceso de revisión.

El proceso de inspección presentado se centra en el paso Preparación, el cual es frecuentemente señalado como el que presenta las mayores debilidades [Parnas 85] [Porter 95]. Acá el paso de Preparación usa una técnica de lectura conocida como "lectura basada en Escenarios" [Porter 95] que proporciona una fuerte sistematización para este paso. La propuesta concentra todo el conocimiento disponible acerca de las propiedades de escenarios en formularios. Cada formulario se diseñó para la detección de un tipo dado de defecto, y se acompaña con guías sobre cómo llenar el formulario y cómo analizar dicha información para capturar el máximo de defectos. La primera versión de esta estrategia de inspección se presentó en [Doorn 98].

La inspección que se reporta fue diseñada para un proceso específico de construcción de escenarios (ver sub-sección 6.1.2), pero es lo suficientemente general para ser adaptada a otros métodos basados en escenarios [Leite 03].

6.2.2. Taxonomía de Defectos

El objetivo de la inspección es garantizar la mejor calidad posible del conjunto de escenarios bajo producción. La flexibilidad y simplicidad de la

estructura de los escenarios permite la existencia de varios defectos, tales como:

- ⇨ Escenarios con información faltante
- ⇨ Escenarios con información errónea
- ⇨ Escenarios con ambigüedades
- ⇨ Escenarios soportando contradicciones
- ⇨ Escenarios parciales o totalmente solapados

Conociendo de antemano el tipo de los defectos, mejora el proceso de detección. Dado que se está tratando con diferentes tipos de defectos, a continuación se presenta una pequeña taxonomía.

Se utilizan muchos términos en relación con los tipos de defectos, como: contradicciones, discrepancias, errores, inconsistencias, ambigüedades, omisiones y conflictos. Pueden agruparse de la siguiente manera:

- ✓ **Discrepancia** comparte su significado con conflicto, contradicción e inconsistencia, que puede bosquejarse como "la presencia de dos o más elementos mostrando características diferentes e incompatibles". Al menos una de las características no es verdadera.
- ✓ **Error** es simplemente una sentencia que no es verdadera.
- ✓ **Omisión** involucra hechos faltantes. Una ambigüedad es un tipo especial de omisión donde el defecto se observa como una falta menor de información, el cual puede evitarse eligiendo una de sus posibles interpretaciones.

Siendo:

D: conjunto de todas las discrepancias
E: conjunto de todos los errores
O: conjunto de todas las omisiones

- ⇨ En esta tesis se denomina defecto a: **D E O**.
- ⇨ Dado que **D E** , una parte de una discrepancia es también un error.
- ⇨ Los términos *defectos* y *DEOs* se usan como sinónimo a lo largo del texto.

Adicionalmente a la clasificación por tipo de defecto, también se usa un criterio de ordenamiento referido a la severidad del defecto: fundamental, organizacional y presentación.

DEOs Fundamentales son aquellos relacionados a la semántica total del escenario. Ejemplos son: falta de un sub-escenario, escenarios superpuestos, incompatibilidad entre título y objetivo, impactos del LEL no tratados por ningún escenario y falta de precondiciones.

DEOs Organizacionales son aquellos relacionados con problemas de sintaxis y de estructura, tales como: episodios incompletos, factorizado de episodios, falta de actores o recursos, falta de un componente en el escenario, actores o recursos que aparecen en episodios pero no están presentes en los componentes Actores / Recursos.

DEOs de Presentación son aquellos que no afectan ni la comprensión ni la consistencia del escenario, por ejemplo, oraciones mal escritas o símbolos del LEL utilizados pero no identificados como tal. Errores de ortografía y gramática no fueron considerados defectos.

6.2.3. El proceso de inspección

El proceso de inspección se basa en la propuesta de Fagan [Fagan 76] con algunos cambios cuidadosamente introducidos para acomodarse mejor al proceso de construcción de escenarios y al producto en sí mismo a inspeccionar.

En esta propuesta, la fase de Apreciación Global ha sido deliberadamente eliminada. En esta fase, los autores deben presentar una visión global de su producto. Dado que el juego de documentos que el inspector recibe (el conjunto de escenarios, el LEL, los formularios e instrucciones) tiene que ser auto explicable, cualquier incomprensión podría ser un indicador de un inconveniente por sí mismo. Suprimiendo la fase de Apreciación Global tiene la ventaja extra de dejar fuera la parcialidad del autor. Esta independencia inicial entre inspector y autores incrementa la relevancia de la Reunión.

Tampoco se tiene la fase de Seguimiento en la estrategia de inspección, pues este paso quedo inmerso en el contexto del propio proceso de construcción de escenarios. La Figura 6-1 muestra la retroalimentación desde Verificar a la actividad Describir.

La fase de Preparación se consagra enteramente a identificar los defectos, en lugar de conseguir conocimiento del material a inspeccionar y se dedica a completar los espacios en blanco en los formularios pre-diseñados, dado que el inspector lee el material para llenar los formularios proporcionados.

Este esquema de actividad se basa en lo que Porter [Porter 95] denomina “lectura basada en Escenarios”, dado que el inspector usa varias técnicas sistemáticas para encontrar diferentes tipos de defectos. Los objetivos de la Reunión son validar los defectos descubiertos en la Preparación con los autores y secundariamente encontrar nuevos defectos.

Como se muestra en la Figura 6-10 el proceso de inspección abarca cuatro actividades: Planeamiento, Preparación, Reunión y Corrección.

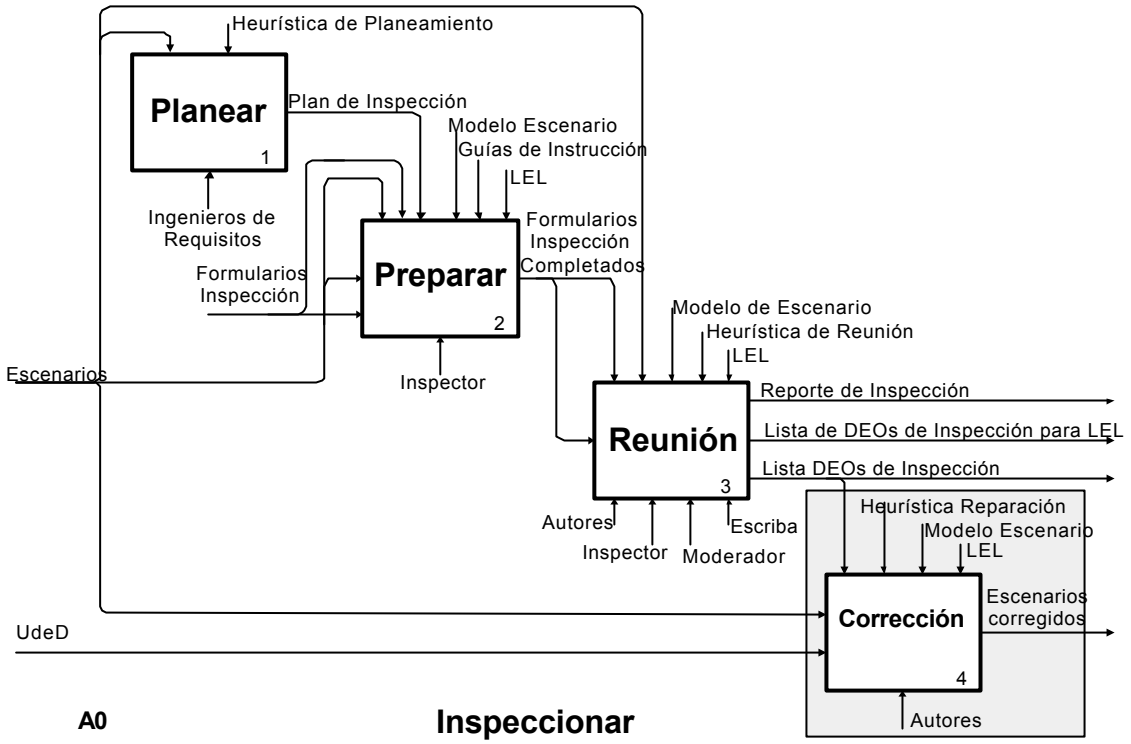


Figura 6-10. SADT del Proceso de Inspección de Escenarios

La actividad Planear consiste en seleccionar el material a inspeccionar (el conjunto de EA entero o parcial), seleccionar los participantes de la inspección, identificar sus roles (inspector, moderador y escriba), preparar y enviar el material (el conjunto de EA, el LEL, los formularios a ser llenados y las instrucciones) al inspector. Los ingenieros de requisitos producen un plan de inspección conteniendo: nombre de los participantes y sus roles, fecha en que se entrega el material al inspector, fecha límite de preparación y fecha estimada de la reunión.

La actividad Preparar apunta básicamente a la detección de defectos por parte del inspector. La preparación consiste primero en leer el LEL para comprenderlo, leer cuidadosamente las instrucciones, y luego completar los formularios de inspección, registrando cualquier DEO descubierto junto con preguntas o dudas presentadas. Las instrucciones contienen guías generales de cómo llenar los formularios y a su vez para cada formulario, cuál es el objetivo de éste, su forma específica de llenado y cómo analizar la información para detectar a través de él ciertos defectos.

Durante la Reunión, el inspector y los autores de los escenarios discuten cada punto detectado en la Preparación mientras el escriba toma nota sobre cada conclusión alcanzada. Cada punto confirmado como defecto o como pregunta sin respuesta se registra en la Lista de DEOs. Eventualmente, durante la Reunión, el inspector o los autores necesitan revisar otras porciones del conjunto de escenarios para apoyar sus puntos de vista, y de esta forma, toparse con otros defectos. El moderador conduce la Reunión para asegurar que todo punto sea tratado adecuadamente y que se cumpla con los objetivos

propuestos. Produce un informe de inspección indicando la disposición final del conjunto de escenarios.

Posteriormente, los autores llevan a cabo la actividad de Corrección. Nueva y más precisa información se elicita del UdeD cuando se responden preguntas y se obtiene una versión nueva mejorada del conjunto de escenarios. Esta actividad no se realiza como parte del proceso de inspección sino que está integrada al ciclo de retroalimentación que involucra las actividades Verificar-Describir en la Figure 6-1. En la Figura 6-10, se muestra la estrategia lo más cercana al modelo de Fagan, por ello, las Figuras 6-1 y 6-10 muestran algún grado de solapamiento, precisamente en la actividad de Corrección.

6.2.4. Intra e Inter inspección de escenarios

La estrategia de inspección de escenarios está realmente compuesta de dos procesos independientes: Inspección Intra Escenarios e Inspección Inter Escenarios, y una actividad de Evaluación. Las actividades Planeamiento, Preparación, Reunión y Corrección se realizan para ambas inspecciones Intra e Inter escenarios. La Figure 6-10 muestra en detalle la actividad Verificar cuya principal salida es la Lista de DEOs. La actividad de Intra Verificación provee una revisión sistematizada de los escenarios individualmente, mientras que la Inter Verificación tiene en cuenta el conjunto de escenarios como un todo. La actividad de Evaluación genera un reporte final de disposición, donde se define el paso siguiente a realizar: una re-inspección después de corregir o la aceptación final. La corrección de escenarios se realiza como parte de la actividad Describir en base a la lista de DEOs producida.

La inspección Intra escenario verifica cada componente en todo escenario para confirmar su consistencia con los otros componentes y la adherencia al modelo de Escenario. La inspección Inter escenario controla la relación entre diferentes escenarios buscando solapamiento y omisiones.

La inspección Intra escenario se lleva a cabo al menos una vez. Cuando la gravedad del defecto incluido en la lista de DEOs lo justifica, una nueva inspección Intra escenario debe requerirse después de la Corrección. La inspección Inter escenario se realiza cuando el conjunto de escenarios ha sido organizado (ver actividad *Organizar* en la sub-sección 6.1.2) y la inspección Intra escenario ha terminado; la inspección Inter escenario se realiza una o más veces de ser necesario.

La Figura 6-11 muestra mediante un diagrama SADT la actividad Verificar en detalle, que se corresponde con la cuarta actividad del diagrama SADT Construir escenarios de la Figura 6-1.

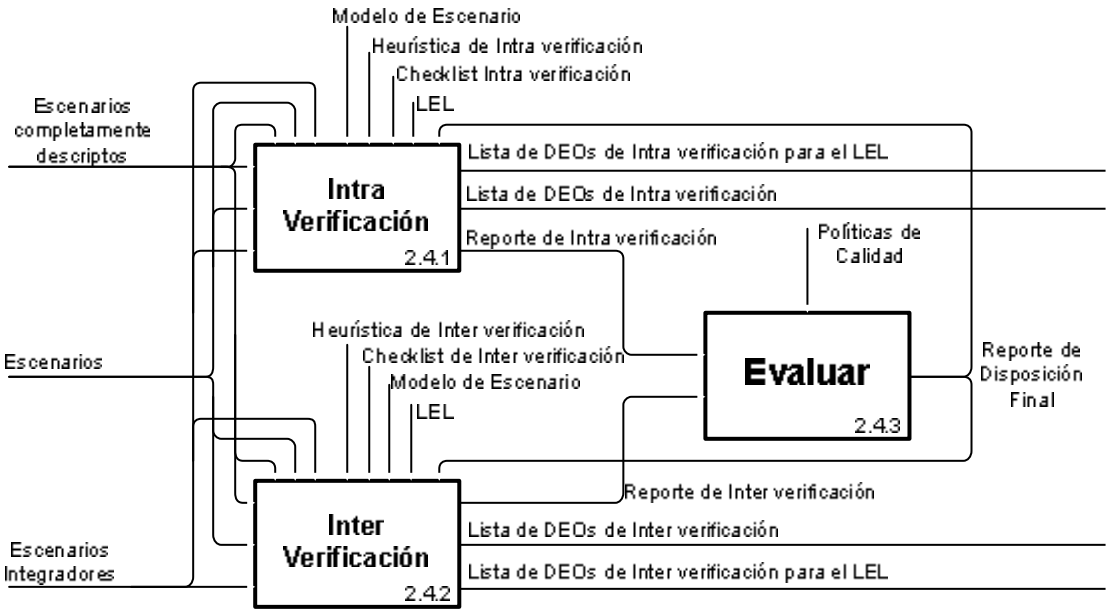
Como ya se mencionó, el proceso de inspección usa formularios pre-diseñados en forma extensiva durante la fase Preparación. Esta es la actividad que consume más tiempo y es la más efectiva para detectar defectos durante la inspección. Cada formulario tiene un juego completo de instrucciones, que especifican cómo deben llenarse los formularios y qué aspecto de los escenarios debe analizarse para capturar los defectos.

Mientras se completa un formulario, el inspector a veces se cruza con defectos cuya detección no es el objetivo en sí mismo del formulario. La detección de defectos no planeados se denomina detección espontánea, y según la experiencia obtenida corresponde a alrededor del 15% del total de defectos descubiertos.

A continuación se presenta un resumen de los formularios junto con algunos ejemplos de formularios con datos obtenidos del caso de estudio "*Sistema de Ahorro del Plan para Adquisición de Automóviles*".

Formularios de Intra inspección de escenarios.

La Inspección Intra escenario es más simple que la inspección Inter escenario pero requiere un esfuerzo mayor. Es más simple dado que cada paso de la comprobación tiene un alcance más acotado y pocos puntos involucrados. El esfuerzo exigido para rellenar estas formas se relaciona principalmente con la gran cantidad de detalles a ser tenidos en cuenta. Los formularios involucrados en la inspección Intra escenario son:



- ⇨ Formas de Control Sintáctico
  - ✓ Comprobación Sintáctica de un Escenario
  - ✓ Comprobación Sintáctica de Episodios
  - ✓ Control Sintáctico de Tipos de Episodios

- ⇨ Forma de Relaciones con el LEL
  - ✓ Cumplimiento del Léxico

- ⇨ Formas de Relación entre Componentes
  - ✓ Control de Ocurrencia de Actores en un Escenario
  - ✓ Control de Uso de Recursos en un Escenario
  - ✓ Comprobación Pragmática de Episodios

- ⇨ Formas de Resumen de Inspección
  - ✓ Carátula
  - ✓ Resumen Cuantitativo de Escenario
  - ✓ Lista Candidata de DEOs
  - ✓ Comentario General

Los formularios para la comprobación sintáctica (ver formas III, IV y IX en la Tabla 6-4) controlan la sintaxis para todos los componentes de un escenario. Hacer que un escenario cumpla con el LEL significa asegurar que los símbolos se usan apropiadamente y que cada frase o palabra identificada como símbolo es realmente parte del LEL (ver la forma VI). Cada sujeto en un episodio debe listarse en el componente Actores y cualquier actor debe jugar un papel en un episodio. Esto se verifica en paso Relación entre Componentes (ver la forma VII). Los recursos y episodios requieren similar control. Una relación semántica entre los componentes es la que existe entre los componentes Objetivo y Episodios. El conjunto de episodios debe cumplir con el objetivo del escenario. Esto también se verifica a través de las formas Relación entre Componentes (ver las formas V y VIII). Los formularios de Resumen de la Inspección (ver las formas I, II, X y XI) recolectan todos defectos descubiertos y las preguntas sin contestar como una guía para actualizar el escenario. Durante la actividad de Corrección ocurre una nueva elicitación en el UdeD.

Una breve descripción de cada formulario de la inspección Intra escenario se brinda en la Tabla 6-4, y en las Figuras 6-12, 6-13 y 6-14 se muestran ejemplos de tres de estos formularios, correspondientes al proyecto “Sistema de Plan de Ahorro para Adquisición de Automóviles”. La Figura 6-15 ejemplifica las instrucciones adosadas al formulario de Intra Inspección VI.

Intra Forma	Descripción
I	**Carátula:** identifica al proyecto, los escenarios a inspeccionar, la versión del LEL, los autores de los escenarios y el inspector. También se registran fechas relevantes, el esfuerzo de construcción de escenarios y el esfuerzo de Preparación.
II	**Resumen Cuantitativo de un Escenario:** da una visión general del volumen de información presente en cada escenario inspeccionado. Mantiene el número de actores, recursos, episodios, símbolos del LEL usados, restricciones y excepciones entre otros valores.
III	**Comprobación Sintáctica de un Escenario:** apunta a verificar si los componentes del escenario, excluyendo episodios, están correctamente escritos. La sintaxis permitida se muestra en el formulario.
IV	**Comprobación Sintáctica de Episodios:** apunta a verificar si los episodios de un escenario están correctamente escritos. Se completa una fila del formulario por cada episodio que no cumple con la sintaxis. Se dispone de columnas para registrar información faltante o excedente en el episodio y en la restricción asociada si la tiene.
V	**Comprobación Pragmática de Episodios:** se usa para detectar episodios irrelevantes o episodios con omisiones de actores o recursos.
VI	**Cumplimiento del Léxico:** apunta a detectar el uso incorrecto de símbolos del LEL en un escenario. Se consideran dos posibles usos incorrectos: a) cuando se usa un símbolo, identificado o no, con un significado diferente del registrado en el LEL, y b) cuando el símbolo es correctamente usado sin ser identificado como un símbolo del LEL.
VII	**Control de Ocurrencia de Actores:** apunta a verificar la coherencia entre las personas y estructuras organizacionales que tienen un rol en el escenario con la lista actual incluida en el componente Actores. Los actores candidatos se comprueban para identificar si son o no símbolos.
VIII	**Control del Uso de Recursos:** es muy similar a la forma VII pues ayuda a controlar la coherencia de los elementos pasivos, medios de soporte y artefactos listados en el componente Recursos contra aquellos usados a lo largo del escenario. También se controla su inclusión en el LEL.
IX	**Control Sintáctico de Tipos de Episodios:** es diseñado para ayudar a la detección de defectos en el uso de caracteres especiales (**# , [,]**) y palabras claves (**si, entonces**) necesarios para representar episodios condicionales y opcionales, o para agruparlos para denotar falta de orden temporal. Ayuda a comprobar la sintaxis del modelo de escenario (ver Figura 5-3), en particular la gramática del episodio[77].
X	**Lista Candidata de DEOs:** está dividida en dos secciones. La primera se utiliza para sugerir correcciones sobre defectos detectados durante la Preparación, y la segunda sección contiene aquellos defectos que requieren información extra del UdeD para repararlos. A cada defecto o pregunta se le vincula el número de formulario y la página origen.
XI	**Comentario General:** da un resumen de la calidad del escenario en función de DEOs detectados, correcciones propuestas y dudas.

Tabla 6-4. Descripción de los formularios de inspección de Intra escenarios

[77] Si esto fuese automático, un parser simple puede detectar la mayoría de estos defectos.

La Figura 6-12 muestra el análisis sintáctico de un escenario, donde por ejemplo, sobra una frase pegada al nombre del actor en el componente Actores y falta indicar la causa de una excepción en dicho escenario.

COMPONENTE	SINTAXIS	FALTANTE	SOBRANTE
Título	Frase | ([Actor | Recurso] + Verbo + Predicado)	---	---
Objetivo	[Actor | Recurso] + Verbo + Predicado	---	---
Ubicación Geográfica	Frase | Nombre	---	---
Ubicación Temporal	Frase | Nombre	---	---
Precondiciones	[Sujeto | Actor | Recurso] + Verbo + Predicado	---	---
Actores	Nombre		*fue aceptado previamente*
Recursos	Nombre		
Excepciones	Causa **(**Solución**)**	*sin explicar causa*	---

Figura 6-12. Forma III: Comprobación Sintáctica de un Escenario

En la Figura 6-13, se lista los símbolos del LEL y su tipo en la clasificación del léxico junto a una referencia del componente del escenario que tiene el problema referido al LEL. En el ejemplo del formulario el término "*Adherente*" se usa en el objetivo y en el segundo episodio pero no se lo identificó como símbolo del LEL, mientras que el término "*Adjudicar por Sorteo*" se lo identificó pero se lo utilizó con una semántica incorrecta en el componente Contexto. En la Figura 6-15 se muestra un instructivo con los defectos que se pueden detectar con este formulario, las indicaciones para completarlo y cómo interpretarlo para descubrir los defectos.

SÍMBOLO	LEL	FALTA DE IDENTIFICACION	MAL USO
Adherente	*Sujeto*	*Objetivo, Episodio 2*	---
Vehículo estándar	*Objeto*	---	---
Adjudicar por Sorteo	*Verbo*	---	*Contexto*

Figura 6-13. Forma VI: Cumplimiento del Léxico

La Figura 6-14 lista palabras que se consideran actores candidatos en el escenario, en qué componentes aparecen y si están correctamente listados en el componente Actores. La última columna del formulario indica si el actor se registra como un símbolo del LEL y su rol en él.

	OCURRENCIA				EXISTENCIA	
CANDIDATO	**TITULO**	**OBJETIVO**	**CONTEXTO**	**EPISODIOS**	**ACTOR**	**LEL**
Adherente	*1*	*1*	*0*	*2, 3*	*Sí*	*Sujeto*
Administrador	*0*	*1*	*0*	*1, 2, 3*	*Sí*	*Sujeto*
Sustituto	*0*	*0*	*0*	*---*	*Sí*	*No*

Figura 6-14. Forma VII: Control de Ocurrencia de Actores en un Escenario

Forma VI. **CUMPLIMIENTO DEL LÉXICO EN UN ESCENARIO**.
Objetivo: Detectar usos incorrectos de símbolos del LEL, tales como uso con un significado diferente al registrado en el LEL, u omisión de destacar el símbolo.
Pasos:

- Registrar en la columna PALABRA / FRASE toda porción de texto destacada como símbolo del LEL, excluyendo los usos duplicados y los sinónimos.
- Para cada una de los presuntos símbolos del LEL, comprobar su existencia en el mismo y registrar en la columna LEL a qué clase pertenece el símbolo involucrado. Si el símbolo no pertenece al LEL indicar también este hecho en el Formulario X.
- Para todos los símbolos del LEL confirmados, inspeccionar el escenario buscando usos del mismo en los que se omitió destacarlos. Si se encontrara alguno registrar el componente en la columna SIN DESTACAR y en el Formulario X.
- Para todos los símbolos del LEL confirmados, inspeccionar el escenario buscando usos del mismo con un significado diferente al registrado en el LEL. Si se encontrara alguno, registrar el componente en la columna MAL USO y en el Formulario X.

Análisis:
Cuando un símbolo del LEL ha sido usado sin destacar y el uso que se dio al símbolo es compatible con el significado registrado en el LEL, la corrección del error simplemente consiste en escribirlo en forma destacada. Cuando el uso de un símbolo del LEL no es compatible con el significado registrado, la corrección del error consiste en reemplazar la mención del símbolo por sinónimos no incluidos en el LEL.

Figura 6-15. Instrucciones para la Intra forma VI: Cumplimiento del Léxico

En el Apéndice F sección F.2 se presentan todos los formularios de la Intra Inspección y las guías de instrucción para su llenado.

Formularios de Inter inspección de escenarios.

La Inspección Inter Escenarios trata con temas más complejos que la Intra Inspección pues maneja todos los escenarios al mismo tiempo. Los formularios involucrados en la inspección Inter escenarios son:

- ⇨ Formas de Relaciones entre Escenarios
  - ✓ Control Cruzado de Escenarios Integradores
  - ✓ Control Cruzado de Precondiciones del Contexto

- ✓ Control Cruzado de Escenarios y Sub-escenarios
- ✓ Control de Precondiciones de Sub-escenarios

⇨ Forma de Solapamiento de Escenarios
- ✓ Control de Solapamiento de Escenarios

⇨ Formas de Cubrimiento del LEL
- ✓ Símbolos del LEL no usados
- ✓ Control de Cubrimiento de Impactos del LEL
- ✓ Control de Ocurrencia de Actores
- ✓ Control de Uso de Recursos

⇨ Formas de Resumen de Inspección
- ✓ Carátula
- ✓ Resumen Cuantitativo de Escenarios
- ✓ Lista Candidata de DEOs
- ✓ Comentario General

Las relaciones jerárquicas entre escenarios y sub-escenarios se verifican mediante las Formas de Relaciones entre Escenarios (ver Formas III, IV, V y VI en la Tabla 6-5). La Forma de Solapamiento de Escenarios (ver Forma VII) trata con límites oscuros y definidos pobremente entre escenarios que comparten porciones comunes. Uno de los defectos más difíciles de detectar son las brechas temporales y se detectan parcialmente en las Formas de Cubrimiento del LEL (ver Formas VIII, IX, X y XI), pues aquellos símbolos nunca usados o poco usados son indicación de posibles gaps entre escenarios. Las Formas de Resumen de Inspección (ver Formas I, II, XII y XIII) recogen todo dato relevante proveniente de las formas anteriores.

La Tabla 6-5 contiene una breve descripción de cada Forma de la Inspección Inter Escenarios. Las Figuras 6-16, 6-17 y 6-18 muestran ejemplos de tres de estos formularios, con datos extraídos del proyecto “Sistema de Plan de Ahorro para Adquisición de Automóviles”.

Inter Forma	Descripción
I	**Carátula:** provee la misma información que la utilizada en la inspección Intra Escenario.
II	**Resumen Cuantitativo de Escenarios:** es muy similar a la usada en la Inspección Intra Escenario. Brinda un panorama general del volumen de información presente en el conjunto de escenarios, tal como cantidad de escenarios por tipo, de actores, de recursos, de episodios, de referencias a símbolos del LEL, de restricciones y de excepciones entre otros valores.
III	**Control Cruzado de Escenarios Integradores:** se usa para verificar que aquellos escenarios mencionados como integradores están correctamente clasificados y descriptos.
IV	**Control Cruzado de Escenarios y Sub-Escenarios:** permite verificar la existencia de sub-escenarios mencionados en episodios o excepciones, e inversamente la clasificación correcta de sub-escenarios nunca mencionados por otros escenarios.
V	**Control Cruzado de Precondiciones del Contexto:** apunta a detectar errores y discrepancias en las precondiciones de escenarios y omisiones de información en otros escenarios que deberían satisfacer dichas precondiciones.
VI	**Control de Precondiciones de Sub-Escenarios:** apunta a detectar errores en precondiciones de sub-escenarios o datos omitidos en aquellos escenarios que menciona a los sub-escenarios, teniendo en cuenta los vínculos jerárquicos establecidos entre escenarios y sub-escenarios y entre escenarios integradores y escenarios.
VII	**Control de Solapamiento de Escenarios:** se usa para detectar la duplicación de información entre escenarios. Para reducir las comparaciones entre escenarios, se calcula un índice de proximidad, basándose en la coincidencia de actores, recursos y contexto. Cuando se obtiene un índice alto, se realizan comparaciones entre objetivos y episodios.
VIII	**Control de Ocurrencia de Actores:** apunta a detectar actores no incluidos como símbolos del LEL pero que tienen una gran participación en escenarios.
IX	**Control de Uso de Recursos:** apunta a detectar recursos no incluidos como símbolos del LEL pero cuya disponibilidad se requiere en varios escenarios.
X	**Símbolos No Usados del LEL:** se usa para detectar omisiones en escenarios mediante el chequeo del nivel de cubrimiento del LEL.
XI	**Control de Cubrimiento de Impactos del LEL:** apunta a detectar situaciones omitidas y discrepancias entre impactos de sujetos del LEL y escenarios / episodios que involucran los correspondientes actores.
XII	**Lista Candidata de DEOs:** provee el mismo tipo de información que la forma usada en la Inspección Intra Escenarios, por ende también se divide en dos seccione. La primera contiene las correcciones sugeridas según los defectos detectados durante la inspección, y la segunda contiene aquellos ítems que requieren información extra del UdeD. Es decir, los DEOs se agrupan en dos conjuntos basados en la posibilidad de resolverlos usando la información disponible. Cada defecto detectado se incorpora indicando la forma y el número de página donde se encontró.
XIII	**Comentario General:** brinda un resumen de la calidad del conjunto de escenarios, basado en los DEOs detectados, las correcciones sugeridas y las dudas emergidas.

Tabla 6-5. Descripción de los formularios de inspección de Inter escenarios

La Figura 6-16 muestra un ejemplo de la Forma VI, que expone las relaciones entre dos escenarios y sus respectivos sub-escenarios, donde la precondición de cada sub-escenario se satisface en este caso por episodios de los escenarios que los mencionan.

ESCENARIO	SUB-ESCENARIO	PRECONDICION	RELACION
8	*9*	*El adherente debe tener los pagos mensuales al día.*	*Episodio 2*
11	*12*	*Hay tantas solicitudes de adhesión como miembros en el grupo.*	*Episodios 1 y 2*

Figura 6-16. Forma VI: Control de Precondiciones de Sub-Escenarios

La Figura 6-17 muestra un ejemplo de la Forma VII, cuyo objetivo es calcular un índice de coincidencia entre dos escenarios. Si el índice es mayor a 0.5, entonces hay una indicación que dichos escenarios deben compararse con mayor detalle. En ese caso, se comparan los objetivos y episodios para determinar si hay un solapamiento o no. Las columnas S_i y S_j de la Figura 6-17 se completan con pares de escenarios y para cada para se calcula la unión de Actores ($A_{\cup ij}$), la cantidad de coincidencias de Actores ($A_{\cap ij}$), la unión de Recursos ($R_{\cup ij}$), la cantidad de coincidencias de Recursos ($R_{\cap ij}$), la unión de sentencias del Contexto ($C_{\cup ij}$) y la cantidad de coincidencias de sentencias del Contexto ($C_{\cup ij}$). El índice de proximidad se calcula para reducir el espacio de búsqueda de pares de escenarios a ser estudiados respecto al solapamiento.

S_i	Sj	ACTOR		RECURSO		CONTEXTO		INDICE PROXIMIDAD	OBJETIVO	EPISODIO	OVERLAP
		$A_{\cup ij}$	$A_{\cap ij}$	$R_{\cup ij}$	$R_{\cap ij}$	$C_{\cup ij}$	$C_{\cap ij}$	$\mathbf{I_{ij}}$			
1	*4*	*2*	*1*	*3*	*0*	*2*	*0*	*---*			
1	*5*	*2*	*1*	*3*	*0*	*2*	*0*	*---*			
1	*6*	*1*	*1*	*3*	*1*	*2*	*0*	*0.33*	*0*	*0*	*---*
…	*…*	*…*	*…*	*…*	*…*	*…*	*…*	*…*	*…*	*…*	*…*
1	*11*	*2*	*1*	*4*	*0*	*2*	*0*	*---*			
1	*12*	*2*	*1*	*2*	*2*	*2*	*0*	*0.50*	*1*	*0*	*---*
1	*13*	*3*	*0*	*3*	*0*	*2*	*0*	*---*			

$$\mathbf{I_{ij} = (\ * C_{ij} + * A_{ij} + * R_{ij}\ ) / (\ * C_{ij} + * A_{ij} + * R_{ij}\ )}$$

Siendo: α , β , γ , factores de peso,

$C \cap_{ij} = |\ Ctx(S_i) \cap Ctx(S_j)\ |$, $A \cap_{ij} = |\ Act(S_i) \cap Act(S_j)\ |$, $R \cap_{ij} = |\ Res(S_i) \cap Res(S_j)\ |$,

$C \cup_{ij} = |\ Ctx(S_i) \cup Ctx(S_j)\ |$, $A \cup_{ij} = |\ Act(S_i) \cup Act(S_j)\ |$, $R \cup_{ij} = |\ Res(S_i) \cup Res(S_j)\ |$,

$Ctx(S_k)$: Contexto del EsScenario k, $Act(S_k)$: Actores del Escenario k,

$Res(S_k)$: Recursos del Escenario k.

Ii,j: es el índice de proximidad del Escenarios i y el Escenario j.

Figura 6-17. Forma VII: Control de Solapamiento de Escenarios[78]

[78] Se debe aclarar que éste es un tema no cubierto porque no se conocen los mejores: α , β , γ debido a que no se avanzó más allá de su postulación.

La Figura 6-18 muestra un ejemplo de la Forma XI donde se muestra un impacto de un sujeto del LEL que es considerado en un determinado escenario mientras que otro impacto del mismo sujeto no está realizado en ningún escenario.

SUJETO	IMPACTO	ESCENARIOS– EPISODIOS	ACTORES
Adherente	*El adherente le paga la cuota mensual al Administrador.*	*Escenario 13, Episodio 1*	*Adherente o Adjudicatario*
Adherente	*El Adherente le da los derechos del plan de ahorro a un tercero de acuerdo a las reglas de cesión.*	---	---

Figura 6-18. Forma XI: Control de Cubrimiento de Impactos del LEL

La Figura 6-19 ejemplifica las instrucciones vinculadas a la Forma VI presentada en la Figura 6-16.

Forma VI. CONTROL DE PRECONDICIONES DE SUB-ESCENARIOS.

Objetivo:

Detectar errores en las precondiciones u omisiones en escenarios que contienen sub-escenarios. Esto se realiza usando los vínculos jerárquicos de los escenarios con los sub-escenarios, y los escenarios integradores con los escenarios.

Pasos:

- Construir una lista incluyendo todos los pares de escenarios vinculados con sub-escenarios. Si el sub-escenario tiene precondiciones, registrarlas en la columna PRECONDICION usando una fila por cada precondición. Si no tiene ninguna precondición, registrar este hecho en la forma XII.
- De la misma manera, agregar a la lista todos los pares de escenarios integradores vinculados con escenarios.
- Determinar la relación entre cada precondición del sub-escenario y precondición del escenario o episodios del escenario ubicados antes de la mención del sub-escenario, y registrarlo en la columna RELACION. Si no se puede establecer ninguna relación, registrar este hecho en la forma XII agregando toda información extra útil.

Análisis:

Cuando el sub-escenario no tiene ninguna precondición, considerar una posible omisión. Dos posibles casos pueden ocurrir:

a) El sub-escenario no es el primer episodio del escenario.
b) El sub-escenario es el primer episodio del escenario.

Si el sub-escenario no es el primer episodio del escenario, revisar los episodios que preceden al sub-escenario. Esta secuencia puede definir parte de la precondición del sub-escenario. Si el sub-escenario es mencionado como el primer episodio del escenario, la precondición del escenario puede ser también la precondición del sub-escenario.

Cuando la precondición del sub-escenario no es satisfecha por un episodio del escenario y no coincide con la precondición del escenario, pueden ocurrir tres casos posibles:

a) Un error u omisión en la precondición del sub-escenario.
b) Un error u omisión en la precondición del escenario.
c) Una omisión de episodios en el escenario.

Cuando ocurre el caso a), sugerir la revisión de las precondiciones del sub-escenario. Si ocurre b), sugerir la revisión las precondiciones del escenario. Bajo la ocurrencia del caso c), sugerir considerar la posibilidad de episodios faltantes o aún de escenarios. Cuando ninguno de estos casos ocurre, debe realizarse una revisión general del escenario y del sub-escenario.

Figura 6-19. Instrucciones para la Inter forma VI: Control de Precondiciones de Sub-Escenarios

En el Apéndice F sección F.3 se presentan todos los formularios de la Inter Inspección y las guías de instrucción para su llenado.

6.2.5. Administrando el Proceso de Inspección

La Reunión es el punto de control en la gestión del proceso de inspección. Durante la Reunión, los participantes desarrollan una visión común del material de inspección. Cada DEO es examinada, se evalúa el enfoque del lector y la calidad total del conjunto de escenarios es sopesada.

Se obtienen dos resultados como consecuencia de la Reunión. El primero es la información requerida para decidir acerca de la disposición final del conjunto de escenarios, es decir, que puedan necesitar corrección y puedan eventualmente requerir una nueva inspección. El segundo resultado es un conjunto de formularios, denominados formas de gestión, completados con información para ser utilizada como retroalimentación al proceso de inspección. Estos formularios se describen en la Tabla 6-6.

Los formularios de gestión se completan en base a la información de la forma Lista Candidata de DEOs. Durante la Reunión, los autores de los escenarios y lectores se juntan con el escriba y el moderador para ratificar o desestimar defectos candidatos. Pueden encontrarse nuevos defectos en el proceso. Cada pieza de información recolectada durante la fase de Preparación se revisa, organiza y evalúa. Se asigna un nivel de severidad a cada defecto, teniendo en cuenta los siguientes valores posibles: fundamental, organizacional y de presentación.

Forma	Descripción
I	**Carátula:** identifica el proyecto, el conjunto de escenarios, la versión del LEL, los autores de los escenarios, el inspector, el moderador y el escriba. También se registran fechas relevantes, cantidad de información obtenida después de la Preparación, el esfuerzo de construcción de los escenarios, el esfuerzo de la Preparación y el esfuerzo de la Reunión. Se anota la disposición final del conjunto de escenarios.
II	**Resumen de DEOs:** muestra un resumen de los formularios. Contiene cuatro tablas donde los defectos se organizan por su tipo (Discrepancia, Error u Omisión), por severidad, por componente y por el modo de detección. Una celda de observación permite al moderador tomar nota acerca de la Reunión y de la calidad del conjunto de escenarios.
III	**Evaluación del Proceso de Inspección:** organiza los defectos por severidad, por el mecanismo de detección y por el formulario de inspección. Los defectos son también clasificados en dos categorías: aquellos cuya reparación pueden realizar los autores del escenario y aquellos que requieren retornar al UdeD. Esta forma se usa principalmente para evaluar el diseño del proceso de inspección.
IV	**DEOs por Componente:** muestra el total de defectos por componente, por tipo de defecto, por tipo de detección y por severidad. La forma se usa principalmente para evaluar los defectos encontrados.
V	**DEOs Rechazados por Forma:** trata con defectos de inspección, inicialmente identificados como defectos del escenario en la Preparación pero descartados en la Reunión. La forma muestra los defectos rechazados por fuente de generación: el proceso o el inspector, y por forma.

Tabla 6-6. Descripción de los formularios de gestión de inspecciones

El moderador indica la severidad del defecto y cómo fue detectado, mientras el escriba registra esta información en los formularios. Si hay alguna duda, se discute con los otros participantes a la Reunión. El moderador resuelve posibles conflictos. Este enfoque permite evaluar el proceso de inspección, la eficiencia del inspector y la calidad total de los escenarios inspeccionados. Mediante la evaluación del proceso de inspección, el proceso mismo puede mejorarse para agilizar la detección de defectos comunes y para intensificar el descubrimiento de defectos ocultos, de manera tal de lograr un proceso más eficiente en esfuerzo y tiempo.

	DISCREPANCIAS		*ERRORES*		*OMISIONES*	
Componente	**ESPONTANEO**	**PROCESO**	**ESPONTANEO**	**PROCESO**	**ESPONTANEO**	**PROCESO**
Título				7, 9		10
Objetivo				2		
Contexto						
Recursos				4, 4		
Actores				4, 9, 10		
Episodios			9	4, 11, 12, 12, 13		10
Excepciones						
Restricciones						
General						

Figura 6-20. Forma IV: DEOs por Componente

Los formularios de gestión descriptos en la Tabla 6-6 se aplican a ambos procesos de Intra e Inter inspección. La Figura 6-20 presenta un ejemplo del Formulario IV con defectos organizacionales del caso de estudio “Sistema de Agenda de Reuniones”, levantados en la Reunión. Los valores en el formulario representan los números de escenarios donde se encontraron los defectos. En el Apéndice F sección F.4 se presentan todos los formularios y sus guías de instrucción.

6.2.6. Observaciones

Se ha presentado una estrategia para la inspección de escenarios a ser realizada por ingenieros de requisitos como un proceso de verificación previo a la validación con el cliente. Esta estrategia fue diseñada para integrarse con los procesos de construcción de escenarios actuales y de escenarios futuros (ver sección 6.1 y capítulo 7) y ha sido aplicada en varios casos de estudio. Es por ello que se puede decir que la inspección de escenarios realmente mejora la calidad de los escenarios. También se ha generado un reporte [Leite 03], en detalle, donde se muestra que esta estrategia puede usarse, con algunas adaptaciones, en otros procesos y con otras representaciones basadas en escenarios.

Aunque la inspección de escenarios no es una idea nueva [Gough 95], la estrategia presentada no se basa en ninguna otra anterior. De la lectura del proceso de Gough se observa que se usó un enfoque Ad-Hoc para la Preparación, mientras que en [Paech 05] se usó la técnica de lectura basada en Escenarios para inspeccionar casos de uso desde la perspectiva de

distintos involucrados, donde el inspector cuenta con descripciones detalladas de las actividades que debe realizar y un conjunto de preguntas que debe responder durante y después de las actividades. En base a los datos de experimentación disponibles [Porter 95] [Basili 96] [Ciolkowski 97] [Fusaro 97] [Kantorowitz 97] [Miller 98] [Sandall 98] [Shull 98] [Porter 98] [Regnell 99c], la estrategia de inspección basada en formularios puede ser más efectiva. Se cuenta con instrucciones precisas para el llenado de los formularios, los cuales ponen en evidencia los defectos, y ayudados con guías de análisis para la detección.

La falta de la fase de Apreciación Global junto con la característica de documentos auto explicativos (conjunto de escenarios, LEL, formularios e instrucciones) evitan cualquier interpretación subjetiva o tendenciosa y enfatiza la importancia de la Reunión. Gente sin conocimiento del UdeD y de los escenarios a inspeccionar puede completar los formularios pero deben estar familiarizados con la representación de escenarios.

Esta estrategia fue madurando a lo largo de los años con varios casos de estudio (ver Tabla 6-8), y esto provee argumentos convincentes de que un proceso de verificación puede y debe realizarse tan temprano como sea posible en el proceso de construcción del software.

Las debilidades del proceso son que es demasiado esquemático y de trabajo intensivo, pero pueden reducirse usando editores inteligentes y agentes de verificación. Sin embargo, no hay que olvidarse que el mayor factor de efectividad de los métodos de inspección reside en que es una persona quien lleva el control del proceso. Las técnicas y herramientas ayudan pero tienen un rol de soporte.

6.3. Resumen

En este capítulo se ha presentado el proceso de construcción de EA, describiéndose en detalle cada una de sus actividades. Se ha expuesto para la verificación de los EA un proceso de inspección basado en formularios, que es una variante de la técnica de Lectura basada en Defectos (ver sub-sección 2.8.3) y que involucra además el proceso de gestión de la inspección.

El proceso de construcción ha sido depurado y mejorado debido a la aplicación del mismo en casos de diverso tamaño tanto en la práctica real como en casos controlados con alumnos de grado avanzados. En la Tabla 6-7 se presentan datos sobre algunos de los casos desarrollados o revisados por la autora, considerando que en cursos de grado se han desarrollado aproximadamente 160 casos entre el 2001 y el 2006 en dos universidades nacionales, llevando a cabo en todos los casos un proceso de inspección.

Año	Caso	Institución	Autores	Cantidad total de Escenarios	Cantidad de Episodios	Cantidad de EI
1995	Sistema Nacional de Emisión de Pasaportes	UB	4 Investigadores	24	186	4
1996	Sistema de Agenda de Reuniones - Versión 1	UB	2 Investigadores	16	85	1
1997	Sistema de Agenda de Reuniones - Versión 2	UB	2 Alumnos de grado	13	63	0
1997	Sistema de Agenda de Reuniones - Versión 3	UB	2 Alumnos de grado	17	63	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 1	UNCPBA	1 Investigador 4 Profesores	22	88	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 2	UNCPBA	4 Profesores	20	57	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 3	UNCPBA	4 Alumnos de grado	16	51	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 4	UNCPBA	3 Alumnos de grado	18	52	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 5	UNCPBA	4 Alumnos de grado	17	76	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 6	UNCPBA	2 Alumnos de grado	8	29	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 7	UNCPBA	3 Alumnos de grado	12	44	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 8	UNCPBA	1 Alumno de grado	20	94	0
1997	Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 9	UNCPBA	3 Alumnos de grado	7	24	0
1999	LEL y Escenarios para los procesos de construcción del LEL y de Escenarios	UNCPBA	2 Alumnos de grado 2 Investigadores	64	268	11
2001	Sistema Contable	UTN-FRBA	2 Alumnos de grado	23	81	4
2001	Sistema de Control de Calidad para Laboratorio Farmacéutico	Empresa	1 Ingeniero de software	42	217	4
2002	Sistema de Campaña de Marketing para Banco	UTN-FRBA	Alumnos de grado	20	52	1
2002	Sistema de Administración de Cajeros automáticos	Empresa	1 Ingeniero de software	20	90	4
2002	Sistema de Gestión de Documentación Electrónica del Juzgado Laboral	UTN-FRBA	3 Alumnos de grado	16	62	1

Año	Caso	Institución	Autores	Cantidad total de Escenarios	Cantidad de Episodios	Cantidad de EI
2004	Planificación y Seguimiento en un Laboratorio Farmacéutico	UTN-FRBA	3 Alumnos de grado	21	136	1
2005	Administración del Almacén de MP y artículos de Encuadernación	UNLaM	3 Alumnos de grado	17	83	3
2005	Seguimiento de Service de equipos para radiocomunicación	UNLaM	2 Alumnos de grado	15	56	1
2005	Sistema de Clasificados de un diario	UTN-FRBA	2 Alumnos de grado	22	85	2
2005	Gestión de Producción de Químicos	UTN-FRBA	2 Alumnos de grado	29	117	5
2005	Administración de Soportes con contenidos de programación por cable	UTN-FRBA	2 Alumnos de grado	19	85	3
2005	Servicio de Ecodoppler y Ecocardiografía	UTN-FRBA	3 Alumnos de grado	19	61	3
2005	Seguimiento de Producción de Cajas de Cartón – Versión 1	UNLaM	2 Alumnos de grado	10	49	1
2006	Seguimiento de Producción de Cajas de Cartón – Versión 2	UNCPBA	1 Alumno de grado	22	74	4
2006	Administración de Préstamos	UNLaM	3 Alumnos de grado	19	99	5
2007	Gestión de convenios universitarios	UNLaM	2 Alumnos de grado	22	72	4

Tabla 6-7. Datos de casos de estudio sobre Escenarios Actuales

En la Tabla 6-8 se pueden apreciar datos referidos a la inspección de escenarios: 7 casos donde se realizó la intra inspección y 2 casos donde además se llevó a cabo la inter inspección. A pesar del gran esfuerzo de inspección, se puede observar que la cantidad de defectos detectados es importante. En [Leite 05] se expone en detalle los datos de la inspección, respecto al grado de severidad de los defectos, al tipo de defectos (discrepancias, errores y omisiones), a la cantidad de defectos detectados durante la preparación y durante la reunión, al grado de detección debido a la aplicación del método, entre otros.

Caso de estudio	Tipo de Inspección	Cantidad de Escenarios	Cantidad de Episodios	Tiempo de Preparación	Tiempo de Reunión	Total DEOs
Sistema de Agenda de Reuniones - Versión 1	Intra	13	56	9 ½ hs.	1 ½ hs.	15
Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 1	Intra	19	92	7 ½ hs.	--	127
Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 2	Intra	12	44	5 ½ hs.	--	65
Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 3	Intra	17	76	15 hs.	--	208
Sistema de Plan de Ahorro para Adquisición de Automóviles - Versión 4	Intra	16	51	9 hs.	--	20
Sistema de Agenda de Reuniones – Versión 2	Intra	17	63	12 ½ hs.	2 hs.	36
Sistema de Agenda de Reuniones – Versión 1	Inter	13	63	6 hs.	1 hs.	7
Sistema de Gestión de Inventario de Planta Cerámica	Intra	45	235	11 ½ hs.	2 hs.	143
Sistema de Gestión de Inventario de Planta Cerámica	Inter	45	235	14 hs.	1 ½ hs.	30

Tabla 6-8. Datos de inspecciones controladas sobre Escenarios Actuales

En la Figura 6-21 se comparan las actividades para construir los EA frente a las actividades generales de todo proceso de definición de requisitos. Se puede observar que Derivar escenarios desde el LEL es una actividad netamente de modelado, mientras que Describir escenarios implica el modelado de los escenarios y la elicitación de información para completar dichas descripciones. La actividad Organizar también involucra el modelado (reorganizar el conjunto de escenarios y generación de escenarios integradores) y la elicitación (recurrir al UdeD en búsqueda de información faltante detectada durante la integración). Finalmente Verificar y Validar son actividades de análisis.

La actividad gestión en la IR involucra en este caso el mantenimiento de versiones de EA con motivo de cambios en los mismos por cambios en el UdeD o mejora en la comprensión de dicho UdeD. Además la trazabilidad implica el mantenimiento de trazas hacia los símbolos del LEL que le dieron origen a los EA. Para una trazabilidad hacia delante se deben mantener trazas a los EF que surgieron a partir de los EA.

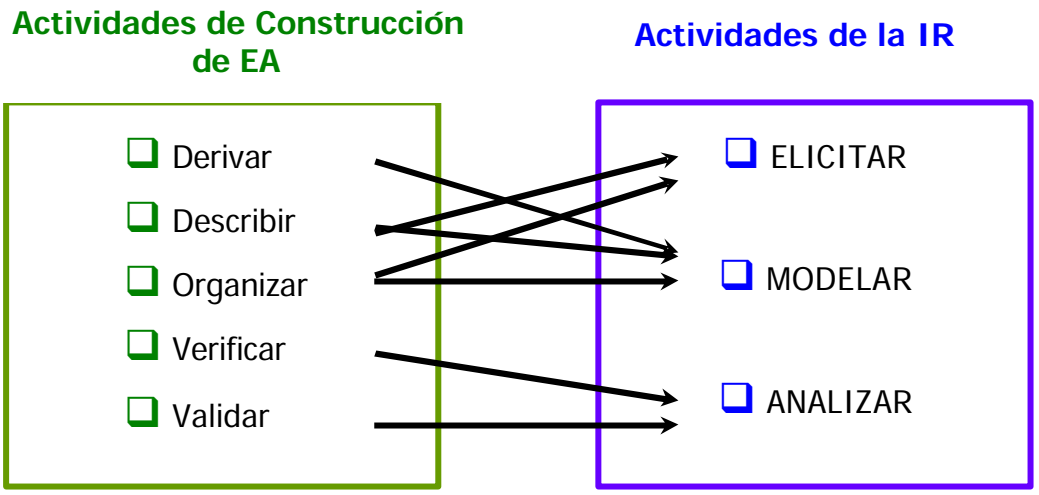


Figura 6-21. Relación entre actividades de IR y actividades de construcción de EA

Parte del material incluido en este capítulo ha sido publicado en [Leite 04b] [Leite 00] [Hadad 99] [Hadad 99b] [Leite 05] [Doorn 98].

Capítulo 7
Escenarios Futuros

«The main purpose of developing scenarios is to *stimulate thinking* about possible occurrences, assumptions relating these occurrences, possible opportunities and risks and courses of action.»

Jarke, Bui and Carroll, "Scenario Management", RE Journal, 1998

7.1. Introducción

Los escenarios futuros (EF) son aquellos que modelan las situaciones proyectadas como solución a los problemas, demandas y necesidades planteadas en el UdeD y de acuerdo a los objetivos propuestos para el sistema [Doorn 02]. Las situaciones observadas son la base para el conocimiento del problema y las situaciones deseadas son el bosquejo para los requisitos. Estos escenarios futuros incorporan naturalmente muchos de los requisitos del software en sus descripciones, siendo la mayor parte de los mismos requisitos funcionales.

Es de esperar que los ingenieros de requisitos presenten más de una visión del futuro, proponiendo soluciones alternativas mediante distintos conjuntos de escenarios futuros (o soluciones alternativas parciales para una o más situaciones futuras). Este proceso no sólo involucra la captura de requisitos directamente de las fuentes de información, sino que requiere una tarea extra "altamente creativa" de los ingenieros de requisitos. En esta etapa, los ingenieros de requisitos comienzan a actuar más como diseñadores y un poco menos como elicitadores.

Los escenarios actuales pueden presentar ya algunos requisitos del software. En la mayoría de las organizaciones existe algún sistema de software de mayor o menor envergadura (desde un sofisticado paquete de software integrado hasta un simple conjunto de planillas de cálculo), por lo tanto, muchos de los requisitos cumplidos por los escenarios actuales serán trasladados inalterados al nuevo sistema de software o, en algunos casos, adaptados o re-creados. Los escenarios futuros naturalmente reflejarán estos requisitos existentes e incorporarán nuevos requisitos elicitados, propuestos y/o negociados durante el proceso de proyección de cómo cambiará el UdeD.

Dado que algunos requisitos del software ya existen actualmente, o ciertos comportamientos del UdeD no se alterarán en el futuro, entonces algunos EA pasarán a convertirse directamente en EF y otros sufrirán alteraciones menores. Es decir, se podrá presentar una correlación 1 a 1 entre algunos EA y algunos EF. En cambio, muchos otros EA serán reemplazados radicalmente y otros directamente desaparecerán. La Figura 7-1 muestra un ejemplo de relación 1 a 1 entre un EA y su evolución en un EF, extraído del

caso de estudio “Sistema Nacional para la Obtención de Pasaportes”.

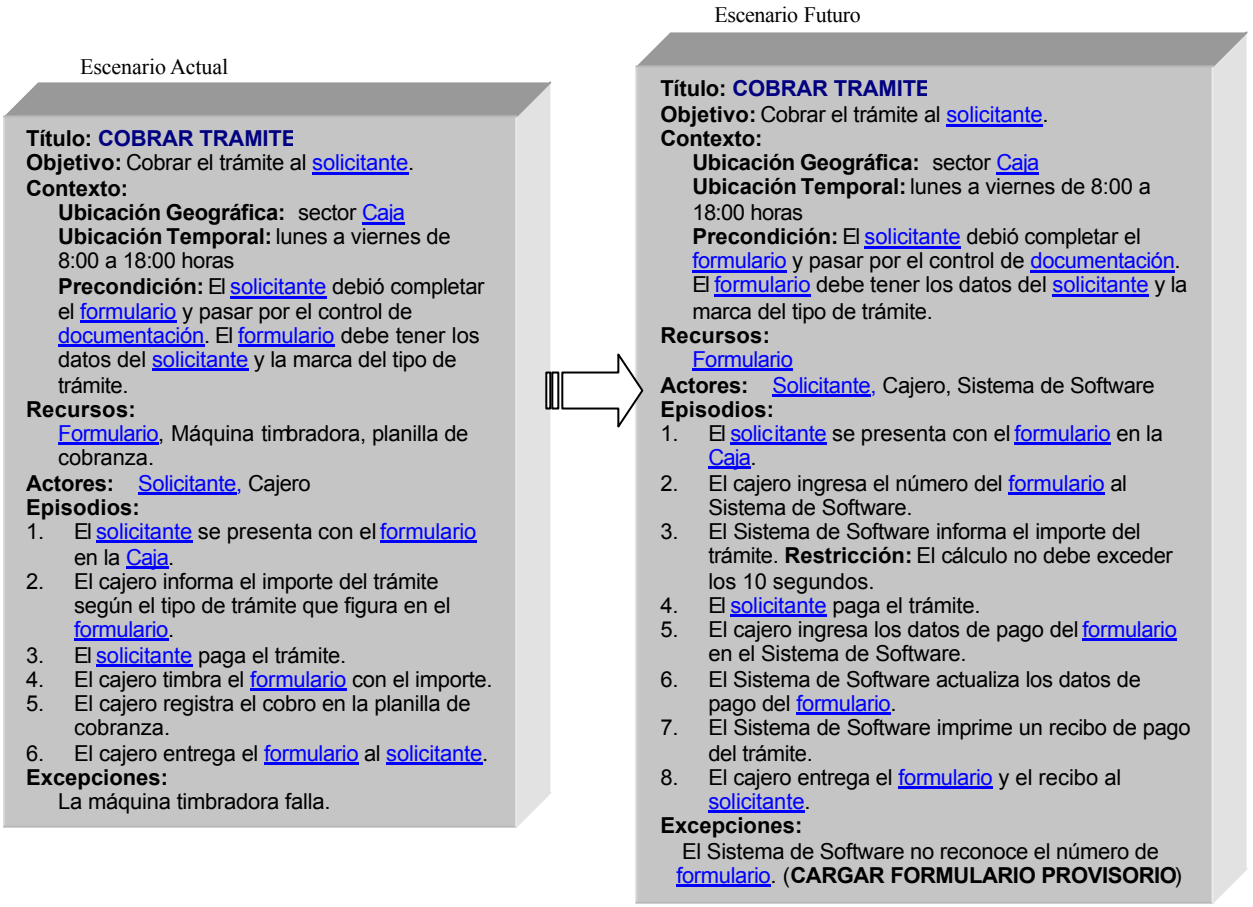


Figura 7-1. Ejemplo de evolución de un escenario actual a un escenario futuro[79]

Dependiendo de las características personales de todos los involucrados y de la naturaleza de las relaciones interpersonales establecidas, los ingenieros de requisitos realizan propuestas relativas al sistema de software, en base a la información adquirida, las cuales pueden mejorar, ampliar y aunque discrepar respecto a las ideas que los usuarios y clientes tienen sobre el sistema de software a construir. Necesariamente esto sienta las bases de una suerte de trabajo en colaboración con algunos componentes de negociación entre partes. Es precisamente durante esta actividad que se define en gran parte el éxito del sistema, ya que el impacto del mismo en el futuro estará fuertemente influenciado por el compromiso de las partes con los objetivos del sistema por un lado y con las concesiones que cada parte haya realizado durante las negociaciones que se lleven a cabo.

El grado de mejoras a implementar en los procesos del negocio es sumamente importante debido a que en situaciones de baja reingeniería, el espacio para que el ingeniero de requisitos pueda hacer propuestas es reducido, ya que se desea introducir un sistema de software en una organización preservando los procesos actuales del negocio en gran medida.

[79] Esta distinción entre EA y EF, utilizando estos términos o similares, ya se encontraban en otros autores [Jarke 98b] [Haumer 99] [Kaindl 00].

Por el contrario, si el sistema de software es concebido como una herramienta para modificar de alguna manera los procesos del negocio, entonces habrá en el futuro una cantidad significativa de actividades que diferirán de las actuales. En estos casos el grado de definición de los procesos del negocio planificados es más pobre, encontrándose frecuentemente situaciones en las que el ingeniero de requisitos se enfrenta con actividades definidas con un nivel de detalle insuficiente, donde se torna imprescindible precisar cómo se realizará la actividad y por lo tanto qué servicios deberán ser provistos por el sistema de software para integrarse a la misma. Este es naturalmente un terreno fértil para que surja espontáneamente una gran cantidad de propuestas por parte del ingeniero de requisitos.

Los requisitos del software son dependientes de los objetivos fijados para el sistema de software. Dichos objetivos son definidos en una primera etapa preliminar del proyecto. En algunos casos pueden ser vagos o imprecisos, pero a medida que el proyecto avanza y se tiene una mejor comprensión del UdeD estos objetivos pueden y deben definirse con mayor claridad. La precisión de los objetivos va acompañada en consecuencia de un ajuste en el alcance del sistema. Este ajuste puede corresponder simplemente a una definición más precisa de los límites del sistema o inclusive a una ampliación o reducción de éstos. Esta evolución que sufren también los objetivos del sistema no sólo obedece a una mejor comprensión de las necesidades del UdeD sino, entre otros, a cambios en la organización o en las políticas internas o externas (por ejemplo, un recorte presupuestario).

Cabe destacar que en general los objetivos del sistema de software son más estables (a largo plazo) y abstractos mientras que los requisitos son más volátiles y más dependientes del contexto. Cambiar un objetivo implicaría cambiar la naturaleza misma del sistema. Cambiar un requisito podría implicar un cambio en la operacionalización de un objetivo [Finkelstein 01].

SDRES no se centra en la elicitación de objetivos sino en descubrir situaciones. Para cada situación se determina su objetivo y por integración se definen objetivos más generales, que corresponden a los objetivos de los escenarios integradores; estos escenarios son los que proveen una visión global de UdeD cuando opere el nuevo sistema de software. Los objetivos de las situaciones que se descubren son los objetivos específicos del proceso del negocio. Con estos objetivos se da contexto a los objetivos inicialmente bosquejados para el sistema de software.

En síntesis, en la etapa preliminar del proyecto se tiene una definición inicial de los objetivos del sistema de software y su refinamiento ocurre en una modalidad bottom-up, teniendo en cuenta que dichos objetivos preliminares sirven como marco de referencia para “crear” los escenarios futuros (ver Figura 7-2). Por lo tanto, el proceso se centra en situaciones guiado por los objetivos preliminares del software, los cuales quedan definidos con precisión una vez finalizado el proceso de construcción de EF en base a los objetivos de los procesos del negocio proyectados. Esta actividad corresponde a la cuarta etapa de SDRES: Definir el contexto del software, produciendo un conjunto de escenarios futuros.

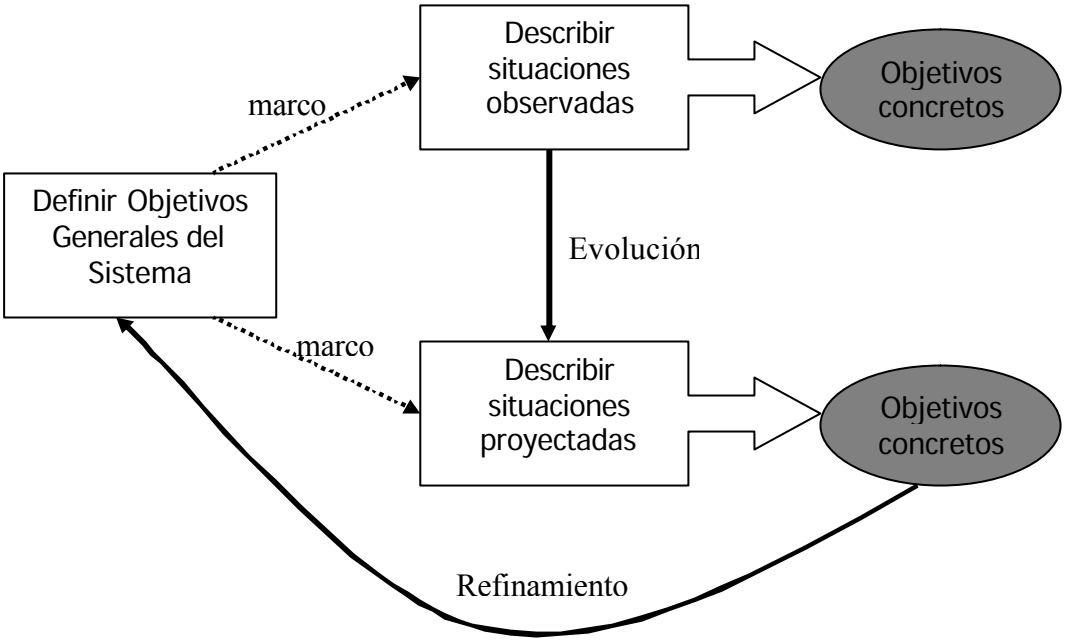


Figura 7-2. Evolución proyectada de Escenarios Actuales a Futuros en el marco de los Objetivos

Como se observa en la Figura 7-2, se distinguen dos tipos de objetivos: los objetivos generales del sistema y los objetivos concretos para cada servicio del software, ya sea el software actual o futuro. Estos objetivos concretos responden a los procesos del negocio, pues los escenarios están describiendo situaciones en el contexto organizacional. Cabe tener presente la distinción entre los objetivos del negocio y los objetivos del software, estos últimos no pueden estar en conflicto con los primeros, pero en algunos casos pueden llegar a alterar sutilmente los objetivos del negocio. Esto dependerá exclusivamente de la aceptación de los clientes. Los objetivos del negocio a veces no están explícitos, pudiendo permanecer ocultos a los desarrolladores y recién aflorar cuando los objetivos expresados del sistema de software entran en conflicto con ellos.

7.2. El proceso de construcción de escenarios futuros

7.2.1. Características del proceso

La primera pregunta que debe contestarse antes de decidir cualquier acción relativa a los EF es:

¿Qué grado de Reingeniería de los Procesos del Negocio se espera?

Cuando el proyecto de software está envuelto en una reingeniería del negocio importante, siendo el sistema un factor clave del proyecto, se espera que muchas situaciones que ocurren actualmente, se modificarán durante la implantación del artefacto de software. Como consecuencia también se espera

que los EF tengan poco apareo con los EA, por ende, el proceso de construcción de los EF debe ser consciente de esta propiedad y ser consecuente con la necesidad de proporcionar un ambiente adecuado para encarar los muchos detalles necesarios para plasmar los objetivos del software.

En el otro extremo, existen algunos proyectos de software que tienen el compromiso de mantener el proceso del negocio tan inalterado como sea posible. Esto puede deberse a varias razones, tales como regulaciones gubernamentales, políticas de la casa matriz, estándares industriales o meramente para preservar una forma exitosa de llevar el negocio. Se espera entonces que haya muchos apareos entre los EA y los EF. Luego, el proceso de construcción de EF debe adecuarse al manejo apropiado de esta peculiaridad.

Los EF de proyectos de software inmersos en un nivel alto de cambios en el proceso del negocio deben construirse en un modo orientado a los objetivos, prestando menos atención a los aspectos procedurales actuales. Por el contrario, proyectos de software con un marco de trabajo de baja reingeniería de los procesos del negocio deben construir sus EF usando un enfoque dirigido por consideraciones procedurales.

Sin embargo, muchos proyectos de software se ubican entre ambos extremos. Para ellos, es necesario combinar ambos enfoques según sus características.

7.2.2. Generalidades del proceso

El proceso de construcción de los EF que se presenta a continuación es similar en ciertos aspectos al proceso de construcción de los EA (ver capítulo 6). La Figura 7-3 muestra el proceso de construcción de los EF, donde las actividades operativas son tres: Seleccionar la estrategia constructiva, Describir los escenarios futuros y Organizar los mismos, mientras que las actividades de análisis corresponden a: Verificar y Validar. Algunas actividades particulares se manifiestan utilizando las mismas heurísticas y siguiendo procedimientos similares a la construcción de EA. Los productos de esta etapa son un conjunto de escenarios futuros y los objetivos del software refinados.

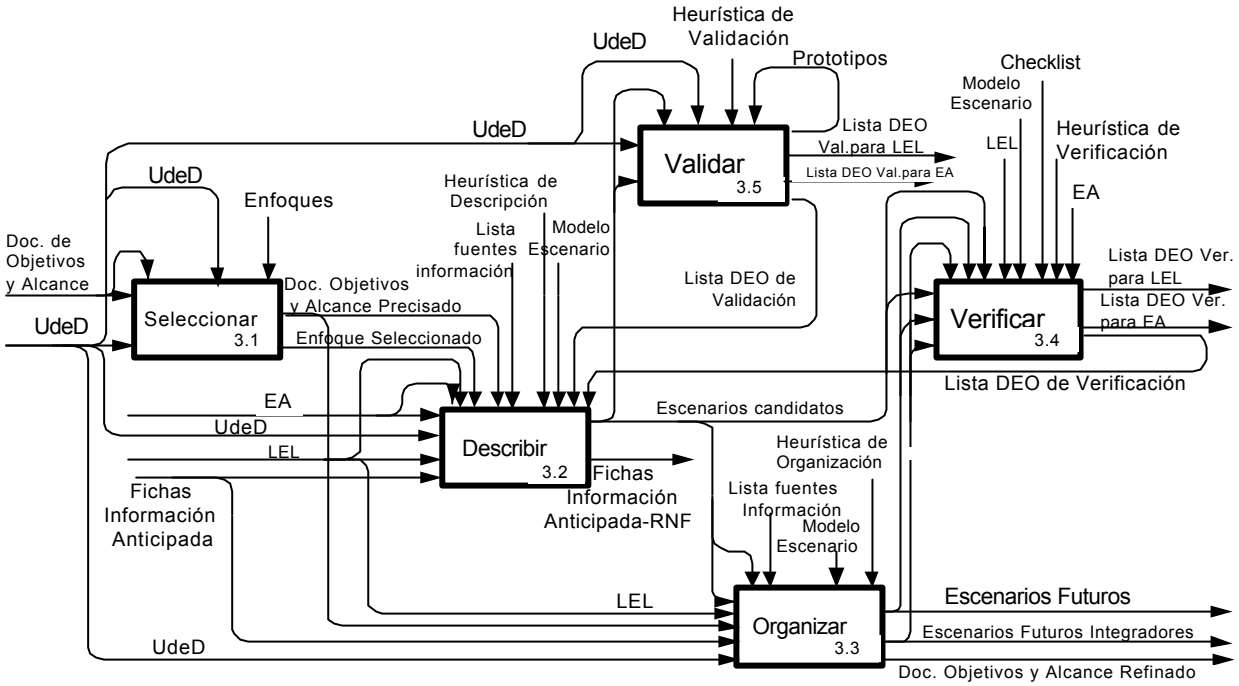


Figura 7-3. SADT del proceso de construcción de Escenarios Futuros

La actividad Describir se basa en estrategias constructivas de EF consecuentes al grado de reingeniería de procesos requerido, pero aplica las mismas heurísticas de descripción empleadas para construir los EA con adaptaciones menores. Las actividades Organizar y Verificar son casi idénticas a las actividades homónimas del proceso de construcción de escenarios actuales (ver capítulo 6), excepto la verificación sobre el cubrimiento del LEL, que no se realiza, y el refinamiento de los objetivos del sistema, que es una nueva tarea dentro de organizar. En el caso de Validar se recomienda otra técnica: la prototipación.

Una vez construidos el LEL y el conjunto de EA, los ingenieros de requisitos cuentan con conocimiento suficiente para comprender el UdeD. Sobre esta base y en función de los objetivos del sistema, comienza la etapa de definición del contexto del sistema de software. Según las características del sistema y el grado de reingeniería de procesos se aplicarán diferentes enfoques para construir los EF.

Durante este proceso, se debe comprender lo que los clientes y los usuarios necesitan y desean para el nuevo software. Por otro lado, se realizan actividades paralelas, como generar propuestas de requisitos, analizar requisitos candidatos elicitados en etapas previas, negociar requisitos y generar soluciones alternativas.

Después de describir los EF, éstos son verificados, corregidos de ser necesario, y luego, organizados. A partir de esta última actividad, se obtienen los escenarios futuros integradores, que dan una visión global del sistema de software y su contexto. Con los escenarios futuros y los escenarios futuros integradores se construye un prototipo, el que se utiliza para validar dicho conjunto de escenarios.

7.2.3. Actividades del proceso

A continuación se detallan las actividades graficadas en el SADT de la Figura 7-3, que son 1) Seleccionar, 2) Describir, 3) Organizar, 4) Verificar y 5) Validar.

(1) SELECCIONAR

El objetivo de esta actividad es seleccionar la estrategia de construcción de los EF. Para ello, se revisan y precisan los objetivos del sistema y se determina el grado de reingeniería de procesos esperado. En base a ello, se opta por un enfoque orientado a los procedimientos, un enfoque dirigido por objetivos o un enfoque híbrido.

Dado el conocimiento adquirido por los ingenieros de requisitos, éstos pueden junto con los clientes y ciertos usuarios de niveles gerenciales mejorar los objetivos del sistema bosquejados durante el planeamiento del proyecto. El grado de mejoras a implementar puede evidenciarse en los objetivos del sistema, aunque es una información que concierne a los objetivos del negocio y deben ser planteados por los clientes. Es decir, en este paso hay una actividad de elicitación que básicamente utiliza la técnica de reuniones.

(2) DESCRIBIR

El objetivo de esta actividad es elicitar y modelar conocimiento sobre las necesidades, demandas, deseos de los clientes y usuarios para el nuevo sistema de software, y/o sobre los problemas, oportunidades y normas no cumplimentadas que se detectan actualmente para solucionar en el futuro.

En esta actividad, se realiza la descripción de los EF siguiendo la estrategia seleccionada en el paso previo (se detalla en la sub-sección siguiente), utilizando como información: los objetivos del sistema de software, los EA, las Fichas de Información Anticipada y la nueva información elicitada sobre el futuro, escribiéndolos haciendo uso del LEL. Durante el modelado de los EF, se realizan varias actividades en paralelo:

- Evaluar los requisitos candidatos que figuran en las Fichas de Información Anticipada para determinar su incorporación a los EF. Estos requerimientos o “requisitos potenciales” se capturaron en etapas previas de SDRES (durante la Comprensión del Vocabulario del UdeD y

la Comprensión del UdeD Actual) y se registraron individualmente en fichas específicas para tal fin.
La información volcada en estas fichas será incompleta, parcial, no verificada, respondiendo posiblemente al punto de vista o intereses de ciertos clientes o usuarios. Es por ello, que en esta etapa, dicha información debe ser evaluada y chequeada para determinar su relevancia y oportunidad frente a los objetivos del software y en función del conocimiento adquirido del negocio.

- Elicitar información sobre lo futuro: necesidades, deseos y dificultades actuales de los clientes y usuarios. Se utiliza como principales técnicas de elicitación: entrevistas estructuradas, cuestionarios y lectura de documentación. Las preguntas básicas se pueden resumir en:

> ***¿Qué espera? ¿Por qué?***
> ***¿Qué necesita? ¿Por qué?***
> ***¿Qué desearía? ¿Por qué?***
> ***¿Qué problemas / dificultades advierte?***

  Esta información elicitada puede volcarse en las Fichas de Información Anticipada para tener la información más organizada y facilitar su uso, o puede modelarse directamente en los EF.

- Elaborar propuestas de requisitos que los ingenieros de requisitos presentan a los clientes y usuarios. Dependiendo de la complejidad o alcance de la propuesta, ésta puede presentarse directamente a los clientes y usuarios, o puede exhibirse a través de uno o más EF. En esta actividad se emplea la introspección como técnica de elicitación.

- Revisar el LEL buscando oraciones tanto en las nociones como en los impactos de los símbolos que presenten un punto de vista formal. Retornando al UdeD, se debe determinar si dichas sentencias deben cumplirse en el futuro. En tal caso debe incluirse dicha información en los EF.

- Negociar ciertos aspectos del software cuyas decisiones son incorporadas a los EF. La negociación permite discutir y resolver conflictos de requisitos y establecer compromisos entre los involucrados. Durante la evaluación de la información contenida en las Fichas de Información Anticipada, los ingenieros de requisitos detectan contradicciones, las que deben presentarse a los involucrados para su discusión y resolución; en otros casos, la información contenida en dichas fichas debe ser confirmada mediante acuerdo. En la mayoría de los casos, la detección de contradicciones, omisiones u otros inconvenientes ocurre al describir los EF. Entonces, la negociación ocurre frente a descripciones de situaciones que son más fáciles de visualizar por los usuarios y, por lo tanto, se facilita la discusión.

La negociación no sólo ocurre por conflictos en los requisitos sino también para evaluar las propuestas de los ingenieros de requisitos frente a determinadas situaciones. Estas propuestas pueden o no estar en conflicto con los requerimientos de los usuarios.
Las reuniones son el modo más efectivo para negociar requisitos y resolver conflictos acerca de los mismos.
Los ingenieros de requisitos encuentran contradicciones, omisiones y superposiciones, y llevan a cabo reuniones con clientes y usuarios que pueden ayudar a resolverlos.

Según Kotonya & Sommerville [Kotonya 98] cada conflicto debe resolverse individualmente, y dividen a la reunión en tres etapas, que adaptándola a los EF bajo estudio consiste en:

i) Una *etapa informativa*: se explican los problemas asociados con un EF.
ii) Una *etapa de discusión*: los involucrados discuten cómo resolver el conflicto. En esta etapa se pueden dar prioridades a los EF[80]. Esto ayuda a decidir qué requerimientos serán eliminados (no serán requisitos del software) y cuáles se incluirán en la especificación del sistema, es decir, serán parte del EF.
iii) Una *etapa de resolución*: se acuerdan acciones a tomar sobre los EF. Las acciones pueden ser:

   - ⇨ eliminar un EF,
   - ⇨ sugerir modificaciones específicas a un EF, o
   - ⇨ elicitar más información acerca del EF.

- ❖ Generar distintas soluciones alternativas, es decir, se construyen distintos conjuntos de EF, o dentro de un único conjunto de EF se presentan alternativas para una o más situaciones particulares. Para ello se basan en distintas propuestas realizadas por ellos mismos, o distintos puntos de vista presentados por los usuarios. Entre las alternativas que se pueden presentar, están por ejemplos distintas maneras de operacionalizar RNF.

En esta actividad, durante la descripción de los EF se aplican las mismas heurísticas de descripción, presentadas en la sub-sección 6.1.2, haciendo la siguiente salvedad respecto al uso del vocabulario descripto en el LEL:

- ✓ Escribir las oraciones maximizando el uso de símbolos del léxico *siempre que esto sea posible*.
- ✓ Los Actores y Recursos deben ser preferentemente símbolos del léxico, *de ser posible*.

Esta salvedad se debe a que en la descripción de los EF pueden surgir términos nuevos debido a nuevas actividades, nuevas acciones, nuevos

[80] Se pueden dar prioridades a los EF o posteriormente priorizar por requisito individual en la siguiente etapa de la estrategia.

recursos, nuevos roles, nuevas condiciones; o pueden no usarse símbolos del LEL por ser obsoletos o carentes de sentido en el contexto una vez éste haya evolucionado. También puede ocurrir que un término se mantenga pero con la noción ligeramente actualizada y probablemente el impacto totalmente alterado, pero manteniendo el significado conceptual. En base a estas consideraciones, debemos hacer notar, que es casi inevitable que el vocabulario del UdeD evolucione junto con la evolución del UdeD mismo.

Por otro lado, se adicionan las siguientes recomendaciones:

- ✓ Describir el qué se deberá hacer, evitar al máximo el cómo se deberá hacer.
- ✓ Evitar incorporar aspectos de diseño, salvo preferencias explícitas de los clientes y usuarios.

Durante esta actividad deben elicitarse RNF del software. Inicialmente, se determinará aquellos RNF que pueden y deben convertirse en acciones del sistema, convirtiéndose en uno o varios RF (operacionalizaciones). Los RNF que persisten como tales se incorporarán como restricciones en los episodios donde interviene el actor sistema de software, y aquellos de carácter más general (no aplicables a un escenario o episodio de escenario) se podrán incorporar a las Fichas de Información Anticipada para ser tenidos en cuenta en la cuarta etapa de SDRES[81]. Básicamente debe cuestionarse en los episodios donde interviene como actor el software la calidad con la que dicho actor debe participar, por ejemplo su eficiencia, confiabilidad y seguridad entre otros. Se pueden elicitar otros RNF en el UdeD siguiendo alguna estrategia específica [Cysneiros 01] [Cysneiros 04] [Kerkow 05] (ver sub-sección 2.6.4) o usando una lista guía de tipos de RNF [Sommerville 02] [IEEE Std 830-1998] [ESA PSS-05-0] [ISO 9126:2001] a modo de checklist.

Cysneiros & Leite [Cysneiros 01] proponen un enfoque para elicitar RNF representándolos en grafos donde los RNF se operacionalizan, y las tareas o los datos identificados se incluyen en los escenarios futuros existentes o se crean nuevos escenarios. Esto implica que muchos RNF desaparecen convirtiéndose en RF empotrados en los escenarios. En algunos casos esto puede implicar tomar decisiones de diseño prematuramente, lo cual es conveniente evitar, salvo que dichas decisiones deban ser tomadas por clientes y no por los diseñadores, ya sea porque involucran políticas organizacionales o preferencias de los clientes y de los usuarios, o decisiones presupuestarias (por ejemplo, invertir en determinado hardware o en un software más sofisticado). Cabe recordar que no todos los RNF pueden operacionalizarse, tal es el caso de algunos requisitos de eficiencia, de portabilidad y de implementación. Kerkow et al. [Kerkow 05] proponen la construcción de modelos de calidad (a partir de modelos de referencia) y checklists para elicitar y documentar los RNF, los cuales luego son incorporados como notas en diagramas de casos de uso o adosados a pasos en las descripciones de casos de uso, dependiendo del tipo de RNF. En SDRES estos RNF se incluirían en

[81] Si se hubiese optado por no realizar la cuarta etapa, se sugiere describir estos RNF como un agregado en el escenario integrador de mayor nivel, o en aquel escenario integrador que tuviera relación directa con cada RNF.

los escenarios integradores o en los escenarios específicos respectivamente, en el atributo Restricción del episodio correspondiente.

Los RNF deben escribirse de tal forma que sean comprobables, sino serán requisitos incompletos o ambiguos. Por ejemplo, *"El sistema debe responder rápidamente frente a las consultas de los clientes"* es un RNF que no puede verificarse en el software. Debería redactarse, por ejemplo, *"El sistema debe responder antes de los 30 segundos frente a las consultas de los clientes en el 90% de los casos, en el resto antes de los 40 segundos"*. No todos los RNF son mensurables, como por ejemplo los requisitos de seguridad pero sí deben ser completos. Por ejemplo, *"El sistema debe mantener seguros los datos de la base"*, es un requisito incompleto pues puede haber muchas maneras de hacerlo y no puede ser comprobado. Debería escribirse, por ejemplo, *"El sistema debe mantener seguros los datos de la base aplicando la técnica de integridad XYZ " o "El sistema debe mantener seguros los datos de la base manteniendo un espejo de todas las transacciones realizadas"*.

Resumiendo respecto a los RNF, éstos se deben elicitar y preferentemente deben buscarse alternativas para su conversión en comportamientos que satisfagan su cumplimiento. Estas alternativas de operacionalización deben evaluarse con los clientes y usuarios, seleccionando la más apropiada e incorporándola a los EF. Deben ser escasos aquellos RNF que permanezcan como tales adosados a los EF. En el ejemplo dado: *"El sistema debe mantener seguros los datos de la base manteniendo un espejo de todas las transacciones realizadas"*, esto involucra inmediatamente su operacionalización y por lo tanto desaparece como RNF. Para el ejemplo dado: *"El sistema debe responder antes de los 30 segundos frente a las consultas de los clientes en el 90% de los casos, en el resto antes de los 40 segundos"*, debe evaluarse con los clientes si esto debe cumplirse siempre o si al ocurrir tiempos excedentes de respuesta el sistema debe reportar el hecho al administrador de la base de datos. En el primer caso, el RNF permanecerá como tal, en el segundo caso implica la operacionalización del requisito en las acciones que debe tomar el sistema. Este es el caso de un RNF de eficiencia que puede o no convertirse en RF según la decisión de los clientes y usuarios. Cabe aclarar para este mismo ejemplo, que si siempre debe cumplirse con los tiempos de respuesta, también puede involucrar la toma de decisión por parte de los clientes en esta fase de la IR, pues los ingenieros de requisitos pueden plantear la necesidad de inversión en hardware, en paquetes de comunicación o en una determinada base de datos, o en producir un software más sofisticado, y son los clientes que deciden, según factores económicos, cronogramas y recursos, cuál opción es la más adecuada para la organización.

Otros escenarios futuros que deben crearse están relacionados con procesos básicos de mantenimiento del sistema de software, como por ejemplo: procesos de Alta-Baja-Modificación de entidades básicas, procesos de depuración y procesos de back-up. Algunos de estos procesos pueden ya estar presentes en EA y sólo requerir una actualización.

(3) ORGANIZAR

Esta actividad utiliza las mismas heurísticas que para la organización de los escenarios actuales (ver sub-sección 6.1.2) y comprende las sub-actividades graficadas en el SADT de la Figura 7-4 y que se describen a continuación.

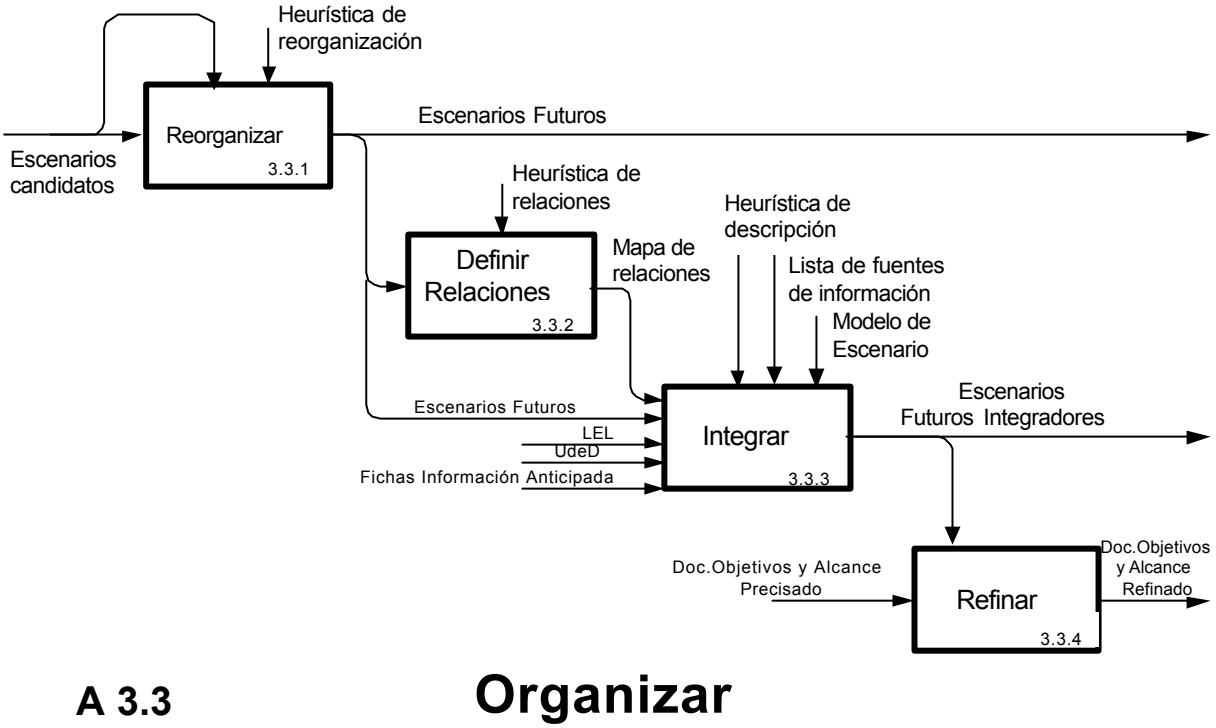


Figura 7-4. SADT de la actividad Organizar en el proceso de EF

- Se aplican las operaciones de composición / descomposición para mejorar la homogeneidad, legibilidad y manejabilidad de los EF. En el caso de una baja reingeniería, es frecuente que haya pocas o nulas operaciones a realizar debido a que el mecanismo constructivo tiende a conservar gran parte de la estructura jerárquica de los EA en los EF.

- Se identifican las relaciones entre los EF: jerarquía, orden, superposición y excepción.

- Se integran los EF teniendo en cuenta las relaciones identificadas previamente, construyendo los escenarios futuros integradores, usando las mismas heurísticas que para los escenarios actuales. Se debe tener en cuenta que en un enfoque orientado a lo procedural puro no sería necesario realizar este proceso, pues los escenarios futuros integradores serían directamente los escenarios integradores actuales.

En la práctica, siempre algún escenario integrador es factible que deba cambiarse, por ende, es factible la realización de la integración.

- ❖ Se refinan los objetivos del software a partir de los objetivos de los escenarios futuros integradores. Esta es una actividad nueva respecto a la construcción de EA, que tiene sentido bajo el concepto de escenario en un contexto organizacional.
  Los objetivos generales del sistema sirven de marco para obtener los objetivos de los EF, por integración se obtienen los objetivos de los EF integradores, a partir de los cuales se refinan los objetivos generales del sistema.

(4) VERIFICAR

Se aplica la técnica de Inspección al conjunto de EF, para determinar su consistencia, completitud y la satisfacción de los objetivos del sistema de software. Se realiza tanto la verificación intra-escenario como la inter-escenarios. Respecto a la verificación contra el LEL, sólo se incluyen los siguientes aspectos:

- ✓ Controlar que todo símbolo del LEL es identificado al mencionarse en un escenario.
- ✓ Controlar el uso correcto de símbolos del LEL. En este caso, puede haber una alteración del significado actual del término debido a la incorporación del software. Esto no implica una actualización del LEL.

En cuanto a la verificación intra-escenario, se adiciona un nuevo control:

- ✓ Controlar que todo objetivo de un escenario esté en concordancia con los objetivos del sistema.

Por otro lado, existe la posibilidad de generar tres tipos distintos de listas de DEOs: una lista para corregir los EF, otra para corregir EA y una tercera menos frecuente para corregir el LEL. Esta última no debe incluir nuevos términos o adaptaciones a significados de símbolos por el hecho de incorporar el software, sólo se incluyen mejoras por una mejor comprensión del vocabulario utilizado actualmente en el UdeD. En cuanto a la lista de DEOs para EA, es posible determinar algún tipo de defecto no detectado previamente, principalmente en el caso de una baja reingeniería, donde hay una correlación más estrecha entre EA y EF, y el defecto detectado en el EF puede también estar presente en el EA.

(5) VALIDAR

Se pueden aplicar distintas técnicas para validar los EF como por ejemplo: reuniones, entrevistas estructuradas, role playing y animación, similares a las aplicadas para validar EA. Pero para el caso de validar EF se recomienda aplicar la técnica de prototipado pues el usuario se encuentra

frente a una descripción de una situación que aún no existe y por ende, debe hacer uso de su imaginación para poder ratificar o corregir dicha descripción. A pesar de estar escritas en LN, respetando al máximo el uso de su propio vocabulario, debe brindarse al usuario una ayuda para la comprensión de estas descripciones.

Se construye un prototipo que será una representación visual de cada escenario futuro, no muestra la interfaz de usuario, sino los pasos que se siguen en dicha situación. Se representa la información de entrada y de salida de cada escenario, las relaciones entre escenarios y los procesos que se llevan a cabo mediante la emisión de mensajes. Durante la ejecución del prototipo de cada escenario frente a los usuarios se anotan, anexo al prototipo, los comentarios que los mismos van realizando.

En caso de tener más de un conjunto alternativo de EF, se deberá construir un prototipo por cada conjunto, de manera tal que se pueda evaluar cada posible solución y elegir la mejor o una combinación de ellas.

Al igual que en la verificación, es posible construir tres listas de DEOs: para EF, para EA y para el LEL.

7.2.4. Enfoques en el modelado de EF

Como ya se mencionó en la sub-sección 7.2.1, la estrategia para encarar el modelado de los EF depende fuertemente del grado de reingeniería esperado en los procesos del negocio donde el software intervendrá. Basados en esta observación de la literatura y la experiencia adquirida en ingeniería de software, se pueden presentar tres estrategias principales para manejar la construcción de los EF:

i) Derivando los EF directamente de los escenarios existentes.
ii) Derivando los EF a través de los objetivos establecidos para el software a construir.
iii) Derivando los EF mediante una combinación de las estrategias precedentes.

Se debe tener presente que si no se construyen escenarios actuales, sólo podrá elegirse el enfoque dirigido por objetivos aunque la construcción de los escenarios futuros no podrá apoyarse en el árbol jerárquico de escenarios actuales. Caben dos alternativas: i) construir escenarios generales futuros capturando conocimiento general del problema bajo la perspectiva de los objetivos del sistema, o ii) derivar situaciones del LEL para considerarlas como escenarios futuros candidatos y evaluarlos frente a los objetivos del sistema. La segunda opción da un punto de partida importante para describir escenarios siguiendo un enfoque middle-out, que es coincidente con el inicio para la construcción de escenarios actuales. La Figura 7-5 muestra un esquema de selección de enfoques considerando la posibilidad de contar con EA o no.

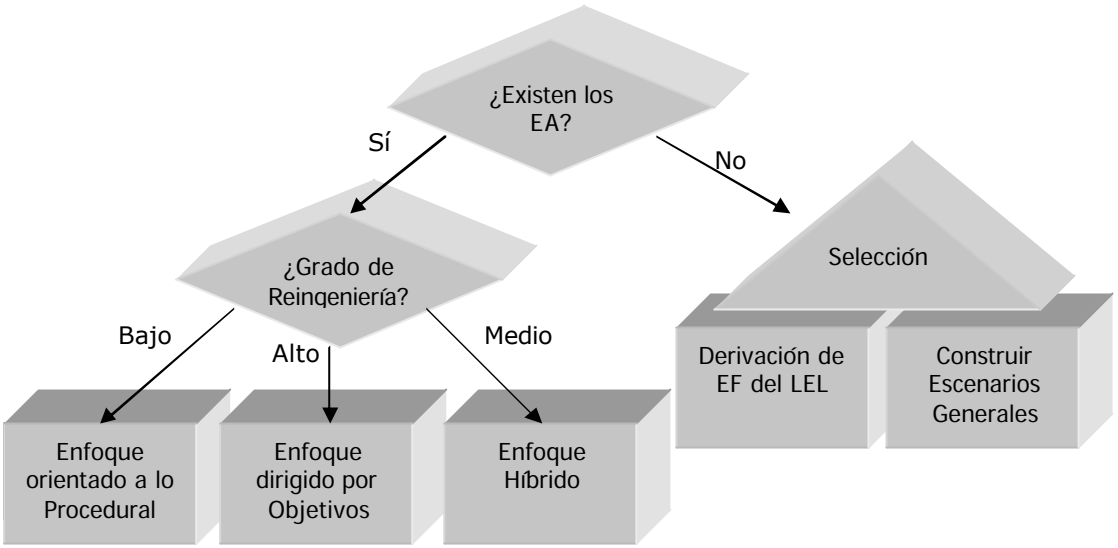


Figura 7-5. Selección de Enfoque para construir EF

Básicamente los EA se estudian en búsqueda de episodios donde hay transferencia de información para evaluar la posible incorporación de tecnología de información que facilite, agilice o haga más eficiente el proceso actual.

A continuación se presentan los tres enfoques para la construcción de los EF que se desarrollan durante la actividad Describir, teniendo en cuenta una ejecución completa de SDRES.

Enfoque Orientado A Lo Procedural

La hipótesis principal cuando se construyen EF en el contexto de procesos del negocio con cambios menores es que hay un mapeo casi completo de los EA a los EF. También se supone que el objetivo del escenario es invariante, es decir, puede copiarse del EA al EF tal como es. Obviamente, ésta es una situación límite, los proyectos reales introducen algunos cambios y no siempre cada EA tiene su correspondiente EF con exactamente el mismo objetivo. Sin embargo, éste es el punto de partida y podría introducirse la desviación a lo largo del proceso. Luego, el EF se construye enfocándose en el componente Episodios del EA. Cada episodio debe analizarse en el marco de los objetivos del sistema de software para determinar si necesita ser modificado y cómo. Una vez que todos los episodios han sido estudiados, el resto de los componentes del escenario debe observarse para propagar los cambios introducidos en los episodios. Pueden necesitarse nuevos recursos o algunos recursos existentes pueden tornarse obsoletos. Si el EF está dentro del alcance del sistema de software, un nuevo actor tiene que ser incluido: el propio artefacto de software. Finalmente, cambios menores en el título y/o el objetivo pueden requerirse.

Aunque cada EF se construye con un enfoque bottom-up, el conjunto completo se construye top-down. El conjunto de EA debe investigarse en el

modo "*post-order*". Esto significa que el primer EA que debe considerarse es el escenario integrador de mayor nivel en la jerarquía. El proceso continúa con el primer escenario integrador del segundo nivel. Luego, la atención se centra en el primer escenario raíz mencionado en el primer escenario integrador del segundo nivel. En el contexto de una realmente baja reingeniería del negocio, se espera alguno o ningún cambio en los escenarios integradores. Debe recordarse que los escenarios integradores no contienen ninguna pieza propia de información. Por lo tanto, los escenarios integradores actúan como marco para ayudar a realizar la auténtica tarea de construcción de los EF, la cual se hace sobre los escenarios raíz y los sub-escenarios. Por ello, se recomienda usar los escenarios integradores actuales sin cambiarlos durante la modificación de los escenarios raíz y los sub-escenarios. Una vez que se tuvieron en cuenta todos los escenarios raíz y los sub-escenarios, deben reconstruirse los escenarios integradores futuros utilizando la misma técnica aplicada en la generación de los escenarios integradores actuales. En el cuadro siguiente se resume el proceso y la Figura 7-6 esquematiza el modo de recorrido de la jerarquía de escenarios actuales.

PROCESO ORIENTADO A LO PROCEDURAL

- Recorrer el conjunto de escenarios integradores actuales usando el modo "*post-order*" (recorrer a partir del escenario integrador actual raíz hacia abajo hasta encontrar un EA, es decir, se evalúan primero las hojas y luego los nodos padres). No alterar los escenarios integradores actuales, serán los escenarios integradores futuros.
- Por cada EA:
    - Determinar si el EA se modificará según los objetivos del sistema.
    - Si no se altera, incluir el EA sin cambio en el conjunto de EF.
    - Si se altera:
        - Examinar los episodios y excepciones del EA en el marco de los objetivos del sistema, y adaptarlos.
        - Actualizar los actores y recursos.
        - Examinar el título, objetivo y contexto desde la perspectiva de las modificaciones introducidas.

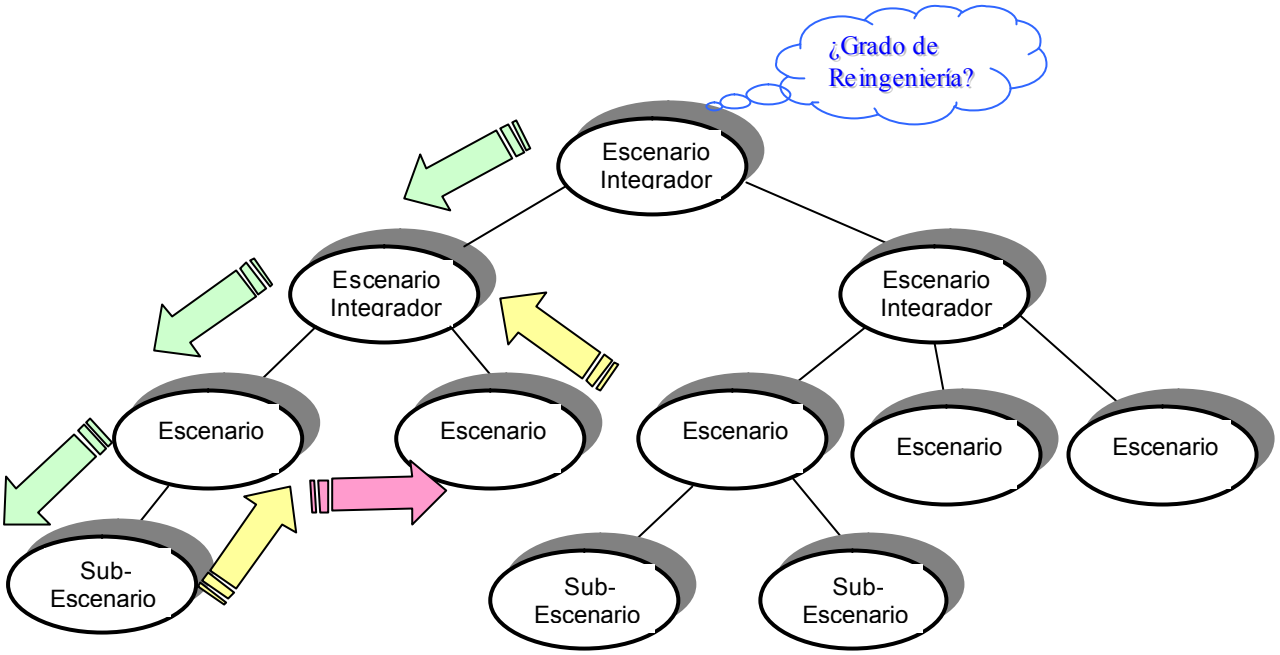


Figura 7-6. Recorrido del árbol en modalidad post-order

Enfoque Dirigido Por Objetivos

La hipótesis principal aquí es básicamente la opuesta al enfoque orientado a lo procedural. Los EA comprendidos por el alcance del proyecto de software serán por lo menos ampliamente modificados. Esto es, escenarios apareados tendrán cambios importantes en los objetivos. También es de esperar que varios EA se volverán totalmente obsoletos y que aparecerán nuevos escenarios.

La tarea en este caso es estudiar el objetivo del EA en el contexto de los objetivos del sistema y determinar cómo éstos modifican al objetivo del escenario. Una vez obtenida una versión preliminar del objetivo del EF, debe determinarse si es posible cumplirlo con un sólo EF o si se necesita más de un EF. También puede ocurrir que el objetivo del EA pierda sentido en el nuevo contexto, tornando obsoleto al EA.

Hasta aquí se han obtenido algunos esqueletos de EF, teniendo una versión preliminar de sus objetivos y sus títulos. Estos escenarios deben completarse sintetizando otros componentes basados en el objetivo.

Cada EF y el conjunto íntegro se construyen usando un enfoque top-down. El conjunto de EA debe investigarse en un modo *"pre-order"*. Aquí, el primer EA que debe considerarse es también el escenario integrador de más alto nivel. En este caso, se esperan muchos cambios en los escenarios integradores. Éstos se denominan escenarios generales y también actúan como marco para ayudar a realizar la tarea efectiva de construcción de los EF, la cual se hace sobre los escenarios raíz y los sub-escenarios. A medida que el proceso avanza, deben aplicarse los nuevos cambios a los escenarios generales para que se mantengan consistentes con los EF ya construidos. Se

aconseja desechar los escenarios generales al final del proceso y construir los escenarios integradores futuros con la misma técnica usada para los escenarios integradores actuales.

Por cada situación en estudio, debe preguntarse:

- ⇨ ¿La situación es compatible con el objetivo del sistema?
- ⇨ ¿El sistema de software jugará un rol o nuevo rol en esta situación?
- ⇨ ¿Es realmente necesaria la situación en el UdeD futuro?

En el cuadro siguiente se resume el proceso y la Figura 7-7 esquematiza el modo de recorrido de la jerarquía de escenarios.

PROCESO DIRIGIDO POR OBJETIVOS

- Recorrer el conjunto de escenarios integradores actuales usando el modo "*pre-order*".
- Por cada escenario integrador actual, construir escenarios futuros generales modificando los escenarios integradores actuales:
  - Revisar el objetivo y título del escenario integrador actual en el marco de los objetivos del sistema.
  - Definir el objetivo y título del escenario general futuro.
  - Actualizar los episodios y excepciones del escenario general futuro (que serán los títulos de los EF).
  - Construir tantos escenarios futuros generales como sean necesarios.
- Recorrer el conjunto de escenarios generales futuros usando el modo "*pre-order*".
- Por cada EA existente (coincide el título del EF con el título del EA):
  - Revisar el objetivo y título del EA en el marco de los objetivos del sistema.
  - Crear tantos EF como sean necesarios, definiendo su título y objetivo.
  - Elicitar o proponer los episodios y excepciones requeridos para cumplir los objetivos de los EF.
  - Actualizar los actores y recursos del EF.
  - Revisar el contexto del EF.
- Describir todo nuevo EF creado (aquél sin apareo con algún EA).
- Construir los escenarios futuros integradores aplicando la técnica de integración (ver sub-sección 6.1.2)

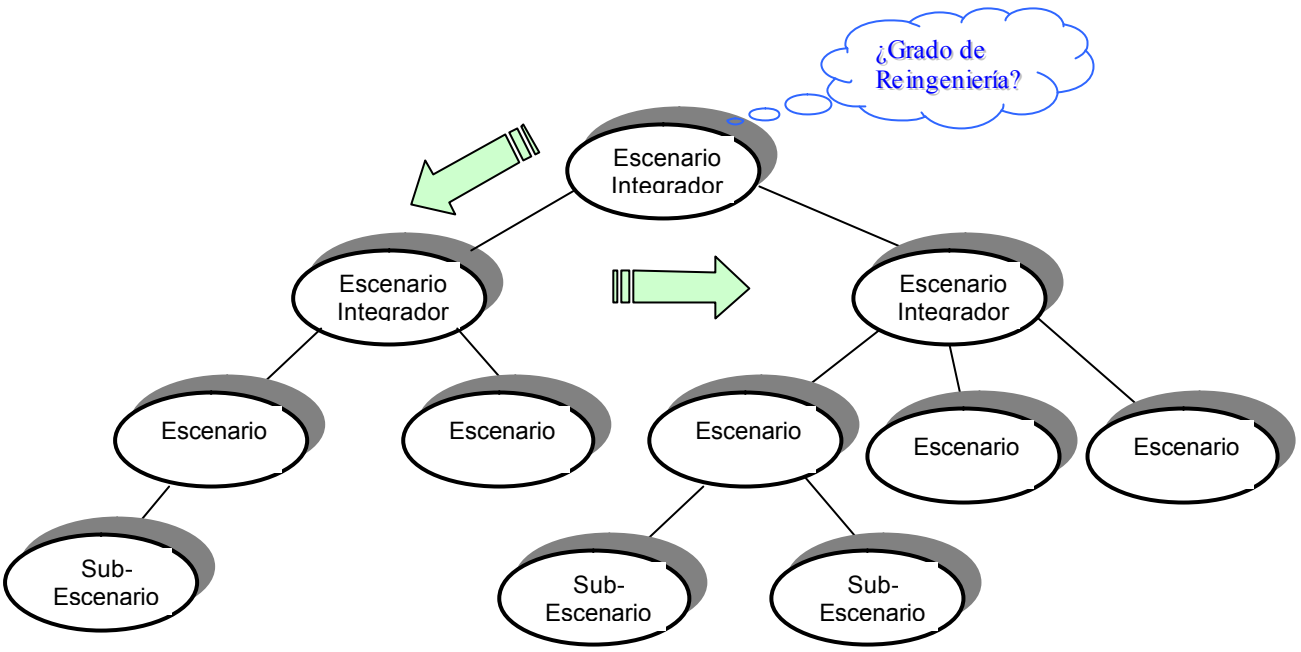


Figura 7-7. Recorrido del árbol en modalidad pre-order

Enfoque Híbrido

Cuando se encaran proyectos de software con un nivel intermedio de reingeniería del negocio, se debe aplicar un enfoque híbrido. Este puede llevarse a cabo reduciendo el alcance en que la pregunta sobre reingeniería debe contestarse. El proceso debe comenzar como si hubiera un nivel alto de reingeniería. De esta manera, los escenarios generales deben obtenerse de los escenarios integradores actuales y cuando el proceso llega a cada escenario raíz o sub-escenario la pregunta: ¿qué grado de reingeniería del negocio se espera? debe contestarse ahora en el alcance del EA bajo consideración. Según la respuesta, se deberá aplicar a este escenario en particular una estrategia orientada a lo procedural o una orientada a los objetivos. Cambiando de una estrategia a otra en cada paso, se puede tratar con todos los posibles grados de reingeniería. Obviamente, un proceso híbrido se reduce a un proceso orientado a los objetivos si la respuesta es siempre “alta reingeniería” para cada EA. Esto no es precisamente verdad si la respuesta es siempre “baja reingeniería”. En este caso la construcción de los escenarios generales y el enfoque “pre-order” induce a realizar algunas tareas completamente innecesarias. Por lo tanto, la pregunta del principio es todavía importante pero el peor precio a pagar por escoger un enfoque híbrido es trabajar más de lo necesario.

En el cuadro siguiente se resume el proceso y la Figura 7-8 esquematiza el modo de recorrido de la jerarquía de escenarios.

PROCESO HÍBRIDO

- Recorrer el conjunto de escenarios integradores actuales usando el modo "*pre-order*".
- Por cada escenario integrador actual, construir escenarios generales futuros modificando los escenarios actuales integradores:
    - Revisar el objetivo y título del escenario integrador actual en el marco de los objetivos del sistema.
    - Definir el objetivo y título del escenario general futuro.
    - Actualizar los episodios y excepciones del escenario general futuro (que serán los títulos de los EF).
- Recorrer el conjunto de escenarios generales futuros usando el modo "*pre-order*".
- Por cada escenario actual existente:
    - Preguntar si es alta o baja la reingeniería para dicha situación específica.
    - Si es alta: seguir una estrategia top-down: (de objetivo a episodios)
        - Revisar el objetivo y título del EA en el marco de los objetivos del sistema.
        - Crear tantos EF como sean necesarios, definiendo su título y objetivo.
        - Elicitar o proponer los episodios y excepciones requeridos para cumplir los objetivos de los EF.
        - Actualizar los actores y recursos del EF.
        - Revisar el contexto del EF.
    - Si es baja: seguir una estrategia bottom-up: (de episodios a objetivo)
        - Examinar los episodios y excepciones del EA en el marco de los objetivos del sistema, y adaptarlos.
        - Actualizar los actores y recursos.
        - Examinar el título, objetivo y contexto desde la perspectiva de las modificaciones introducidas.
- Por cada nuevo EF seguir una estrategia top-down
    - Describir todo nuevo EF creado (aquél sin apareo con algún EA).
- Construir los escenarios futuros integradores aplicando la técnica de integración (ver sub-sección 6.1.2)

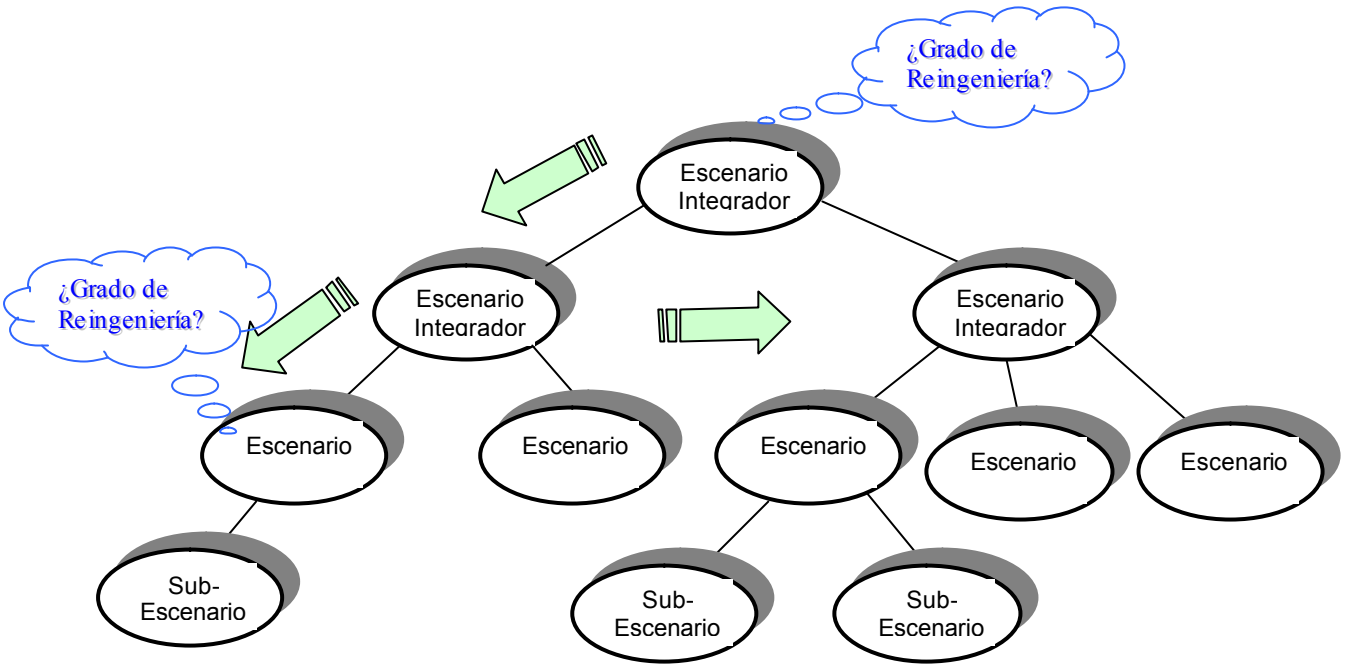


Figura 7-8. Recorrido del árbol en modalidad híbrida

7.3. Observaciones

En esta tesis, se enmarcan los escenarios en un contexto organizacional [Leite 00], el cual puede ser el contexto presente o el contexto proyectado. Mientras que los métodos y estrategias para capturar requisitos basadas en escenarios o casos de uso propuestas por otros autores [Jacobson 95], [Johnson 95], [Hsia 94], [Regnell 95], [Kaindl 00] se centran en describir la interacción entre los actores del UdeD y el sistema de software. Esta tesis, por el contrario, enfatiza el estudio del contexto en el cual el sistema de software quedará inmerso [Leite 04], pues considera que la solución más adecuada a un problema es intrínsecamente dependiente del contexto en el que dicha solución será aplicada. Se comparte con Haumer et al. [Haumer 99] la idea que la visión del sistema futuro está fuertemente influenciada por el contexto del pasado del sistema.

Numerosos autores combinan escenarios con objetivos, pero son pocos quienes reconocen la necesidad de conocer el contexto actual, y varios limitan el conocimiento a la interacción entre usuarios y el nuevo sistema. Se menciona a continuación algunas de estas propuestas.

En la propuesta de Axel van Lamsweerde [van Lamsweerde 98] se elicitan en paralelo objetivos y escenarios del sistema deseado. Los procesos de elaboración de objetivos y de elaboración de escenarios se entrelazan. La descripción de un escenario concreto puede llevar a elicitar especificaciones de objetivos y viceversa. Los objetivos se refinan top-down y se abstraen bottom-up, y paralelamente se elaboran escenarios.

La técnica de acoplamiento bi-direccional de objetivos y escenarios para obtener requisitos propuesta por Rolland et al. [Rolland 98] [Rolland 99] comienza por la identificación inicial de objetivos, y luego, repite el proceso de analizar los objetivos, crear escenarios para concretarlos y explorar éstos para elicitar objetivos. A diferencia de [van Lamsweerde 98], ésta es una técnica incremental que repite la secuencia descubrimiento de objetivos – creación de escenarios.

La estrategia presentada por Colin Potts [Potts 95] se centra en objetivos mediante un proceso top-down que identifica objetivos generales del dominio y los descompone en una jerarquía de objetivos y sub-objetivos hasta su operacionalización mediante escenarios.

La propuesta de Hermann Kaindl [Kaindl 00] relaciona objetivos, escenarios y funciones entre el sistema actual y el nuevo sistema, de manera tal que se pueda rastrear la evolución de cómo se ejecuta una tarea. Prescribe secuencias parciales a seguir según la información conocida en determinado momento: objetivos conocidos o escenarios conocidos o funciones conocidas. Este autor observa la necesidad de conocer lo actual y utiliza dicha información para el sistema a construir.

Tanto [van Lamsweerde 98] como [Rolland 99] construyen escenarios futuros y objetivos futuros desde cero sin manejar escenarios actuales, mientras que [Kaindl 00] tampoco construye un modelo del UdeD actual pero recolecta escenarios conocidos usándolos para crear escenarios futuros, aunque su propuesta no trata adecuadamente el problema de completitud.

En esta tesis, a diferencia de [Potts 95], se usan los objetivos del sistema de software para guiar y ayudar en parte del proceso de construcción de escenarios futuros. La construcción de los EF interactúa fuertemente con los objetivos del sistema de software. Éste es un camino bi-direccional pues, por un lado, los objetivos del sistema afectan el cómo se construyen los EF y, por otro lado, una vez finalizada la descripción de los EF, el conocimiento sobre los objetivos del sistema ha aumentado, dado lo cual se puede refinar la definición de los mismos.

Cabe recordar que las situaciones observadas (plasmadas en EA) son la base para el conocimiento del problema y las situaciones deseadas (plasmadas en EF) son el bosquejo para los requisitos. Se debe tener claro que la situación deseada es en determinado instante la situación actual cuando se implementa un cambio en el sistema de software. La diferencia entre escenarios actuales y escenarios futuros es válida solamente bajo la visión que se parte de escenarios que describen el macrosistema a escenarios que describen cambios en él.

Por otro lado, en la literatura hay pocos autores que tratan los RNF en los escenarios y casos de uso. RUP [Jacobson 99] incorpora requisitos de rendimiento y de fiabilidad en los casos de uso, y para el resto de los RNF genera una lista adicional, pero no aclara cómo realizar la captura de los mismos, sólo su modelado.

7.4. Resumen

En este capítulo se presenta un proceso de construcción de EF, cuyo enfoque dependerá del grado de reingeniería de los procesos del negocio. En esta fase es fundamental el acuerdo de los clientes y usuarios respecto de la funcionalidad y condiciones bajo las que el sistema de software va a operar. Para lograr una mejor validación de los EF se aconseja utilizar la técnica de prototipado, mediante la cual podrá mostrarse alternativas para los procesos del negocio utilizando el sistema de software.

Los EF son construidos desde el punto de vista del proceso del negocio, incorporando las acciones del actor "sistema de software". Es así, que el mundo descripto por estos escenarios no es observable sino que describen la forma en que se espera que se desenvuelva el negocio cuando se disponga del sistema de software que se planifica desarrollar. No toda la información incluida en los EF está directamente relacionada con el sistema de software, por el contrario muchas de ellos describen situaciones en las que no participa el sistema de software. Estas situaciones se constituyen en una eficaz descripción del contexto en el que el sistema se habrá de desenvolver.

Dado que los EF contienen todos los requisitos funcionales pero no hacen énfasis en los requisitos no funcionales, se sugiere la aplicación de alguna estrategia que ayude a su captura y modelado. Gran parte de estos RNF se pueden reflejar en los EF, ya sea mediante operacionalizaciones o su inclusión en el atributo Restricción aplicable a los episodios de los EF.

El proceso descripto en este capítulo ha sido aplicado a diversos casos desde el 2002 a la fecha, realizados por la autora y por estudiantes de grado avanzados con supervisión de la misma. En la Tabla 7-1 se exponen algunos de estos casos, considerando que se han realizado entre estos años aproximadamente 125 casos de estudio en dos universidades nacionales.

Año	Caso de Estudio	Institución	Autores	Cantidad total de Escenarios	Cantidad de Episodios	Cantidad de EI
2002	Sistema de Administración de Cajeros automáticos	Empresa	1 Ingeniero de Software	38	195	4
2004	Planificación y Seguimiento en un Laboratorio Farmacéutico	UTN-FRBA	3 Alumnos de grado	20	142	1
2005	Administración del Almacén de MP y artículos de Encuadernación	UNLaM	3 Alumnos de grado	17	86	3
2005	Seguimiento de Service de equipos para radiocomunicación	UNLaM	2 Alumnos de grado	12	55	1
2005	Sistema de Clasificados de un diario	UTN-FRBA	2 Alumnos de grado	23	94	2
2005	Gestión de Producción de Químicos	UTN-FRBA	2 Alumnos de grado	29	171	5
2005	Administración de Soportes con contenidos de programación por cable	UTN-FRBA	2 Alumnos de grado	20	81	3
2005	Servicio de Ecodoppler y Ecocardiografía	UTN-FRBA	3 Alumnos de grado	22	103	5
2005	Seguimiento de Producción de Cajas de Cartón – Versión 1	UNLaM	2 Alumnos de grado	10	58	2
2006	Seguimiento de Producción de Cajas de Cartón – Versión 2	UNCPBA	1 Alumno de grado	8	43	2
2006	Administración de Préstamos	UNLaM	3 Alumnos de grado	20	103	5
2007	Gestión de convenios universitarios	UNLaM	2 Alumnos de grado	22	91	4
2007	Servicio de Presupuesto y Seguimiento de Pedidos	UNLaM	2 Alumnos de grado	38	171	5
2007	Sistema de Agenda de Reuniones – Versión 1	UNLaM	1 Investigador	17	90	1
2007	Sistema de Agenda de Reuniones – Versión 2	UNLaM	1 Investigador	24	112	2

Tabla 7-1. Datos de casos de estudio sobre Escenarios Futuros

Se observa en la Figura 7-9 que: la actividad Seleccionar la estrategia implica elicitar objetivos; Describir los EF involucra actividades de elicitación, modelado y análisis donde hay una fuerte actividad de negociación; Organizar los EF involucra el modelado al aplicar operaciones sobre los escenarios y construir los escenarios integradores, y la elicitación de información omitida detectada durante la integración; Verificar y Validar son actividades de análisis, donde al validar puede ocurrir la elicitación como una actividad secundaria.

La gestión en la IR involucra el mantenimiento de versiones de EF y la trazabilidad de estos EF a los EA que le dieron origen, como también trazas hacia la especificación de los requisitos que los EF contienen.

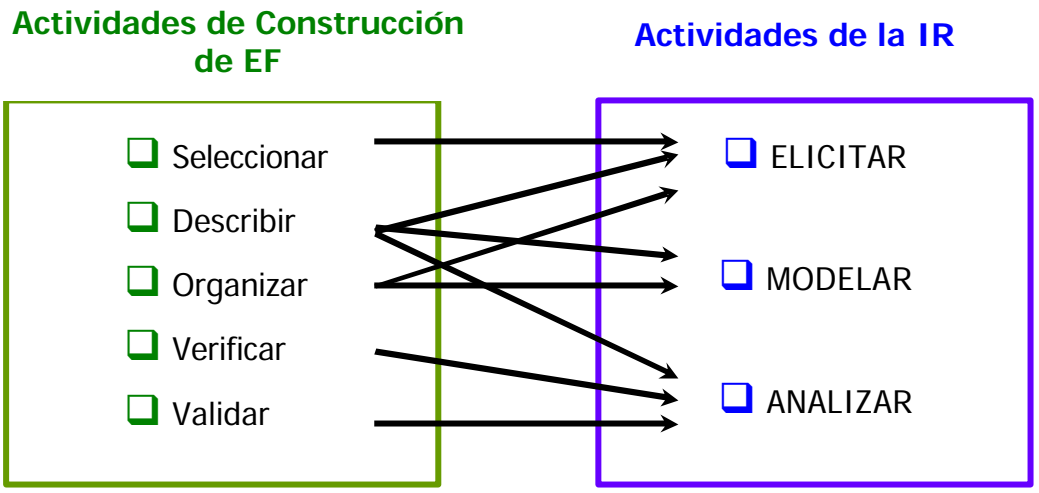


Figura 7-9. Relación entre actividades de IR y actividades de construcción de EF

Parte del material incluido en este capítulo ha sido publicado en [Leite 04b] [Doorn 02] [Hadad 03].

Capítulo 8
Explicitar Requisitos del Software

«The primary output (of a requirements process) is our collective understanding of the customer's problem. The specification is only a representation, a model of that understanding»

B. Lawrence, "Designers must do the Modeling", IEEE, 1998

8.1. Introducción

Una vez construidos los escenarios futuros, los requisitos del sistema de software están fácilmente disponibles. Los escenarios futuros no son requisitos pero los contienen en distintos componentes del escenario, de donde pueden ser extraídos con cierta facilidad. A veces es deseable individualizar los requisitos y plasmarlos en un documento.

Muchas de las buenas prácticas o prácticas recomendadas en el proceso de desarrollo de software se basan en la existencia de un documento de requisitos en el que se individualizan los mismos en forma precisa. Esta individualización facilita la rastreabilidad de los mismos, el control de sus cambios, el análisis de su interdependencia, el apareamiento de los requisitos con los componentes de software, el seguimiento de la evolución de los requisitos y el versionado de los mismos, entre otras actividades.

La evolución de los requisitos (ver sub-sección 1.1.4) es un factor crítico en un proyecto de software, entonces la gestión de requisitos es una actividad fundamental en el desarrollo y el mantenimiento posterior del software. Explicitar los requisitos empotrados en los escenarios futuros es una forma de contribuir a la gestión de requisitos, promoviendo la transparencia en los cambios de éstos.

Por otro lado, muchas organizaciones solicitantes de un software estipulan la necesidad de un documento que avale todos los requisitos del software como parte del contrato con la empresa desarrolladora. Es frecuente que este documento sea utilizado como parte del pliego de licitación o de un anexo técnico en el caso de sistemas de software de cierta envergadura. Se puede decir que la generación del SRS es una actividad dirigida para los clientes, mientras que la construcción de los escenarios futuros permite la elicitación y validación de requisitos, y como producto final son de gran utilidad para el diseño del software.

Los EF incorporan muchos de los requisitos del software en sus descripciones, especialmente aquellos requisitos de naturaleza funcional. Un aspecto a destacar es que la calidad de los requisitos obtenidos de los EF está garantizada pues ellos han sido previamente verificados, validados y

acordados, y además estos requisitos no presentan conflictos.

En general, se puede decir que muchas estrategias publicadas parten de un documento o lista de requisitos, cuyo origen no es explicado, A partir de la misma se resuelven conflictos, se dan prioridades, se gestionan cambios en los requisitos y se aparean con los componentes del software a desarrollar. Es decir, existen muchas actividades de gran importancia en el desarrollo del software que están motivadas a partir de un conjunto de requisitos cuya elicitación y definición no están descriptas ni justificadas, y donde dicho conjunto pareciera surgir espontáneamente en el proceso de desarrollo.

8.2. El proceso de explicitar los requisitos

En la cuarta etapa de SDRES, se modelan los requisitos del sistema de software en un documento. Este proceso comienza buscando requisitos funcionales y algunos no funcionales en los escenarios futuros. La gran mayoría de los RF pueden ser recolectados de los episodios donde el sistema de software es un actor involucrado. Los RNF pueden capturarse de las restricciones y excepciones de los escenarios futuros. Además, se revisan todas las Fichas de Información Anticipada no consideradas en la etapa anterior, o creadas en dicha etapa, para detectar RNF que no han sido vinculados a ningún escenario futuro.

La lista de requisitos no requiere ser verificada ni validada dado que se construye a partir de un conjunto de EF que ha pasado por procesos de V&V. Además se supone que en esta etapa los requisitos no necesitan ser acordados porque ello ya fue realizado a través de los EF. Si los EF no fueron priorizados, entonces es probable que se realicen reuniones con los clientes y usuarios para dar prioridades a los requisitos.

Se redacta el Documento de Definición de Requisitos, basándose en algún estándar que la organización utilice o que se proponga de acuerdo al proyecto en particular. Cabe mencionar que este documento respeta el vocabulario del UdeD, manteniendo vínculos al LEL, pues por un lado fue generado a partir de los EF descriptos utilizando el LEL y todo otro agregado debe realizarse respetando el uso de dicho vocabulario. Luego el SRS es un documento de fácil lectura para los clientes y usuarios, y que además presenta nula o escasa ambigüedad.

Debe destacarse que se está partiendo de un conjunto de EF, el cual no sólo fue verificado y validado, sino que los RNF elicitados fueron muchos de ellos operacionalizados. La operacionalización de RNF implica la inclusión de nuevas acciones en los EF. Otros RNF permanecieron como tales en los EF, por implicar decisiones de diseño o propiedades intrínsecas del software, y fueron adosados a escenarios o episodios de escenarios utilizando el atributo Restricción. Cuando se dice RNF adosados a escenarios, esto indica que el RNF es incluido en la Restricción de un episodio de un escenario integrador.

Las actividades involucradas en el proceso, que se presenta mediante

un modelo SADT en la Figura 8-1, son: 1) Generar la lista de requisitos, 2) Priorizar los requisitos, 3) Describir los requisitos en un SRS y 4) Verificar el SRS.

Este proceso no requiere una validación pues toda la información que se maneja en él proviene de una fase anterior ya validada, excepto la asignación de prioridades, pero esta actividad se logra por consenso en reuniones. Mientras que sí se requiere una verificación para comprobar la correcta explicitación de requisitos y generación del SRS.

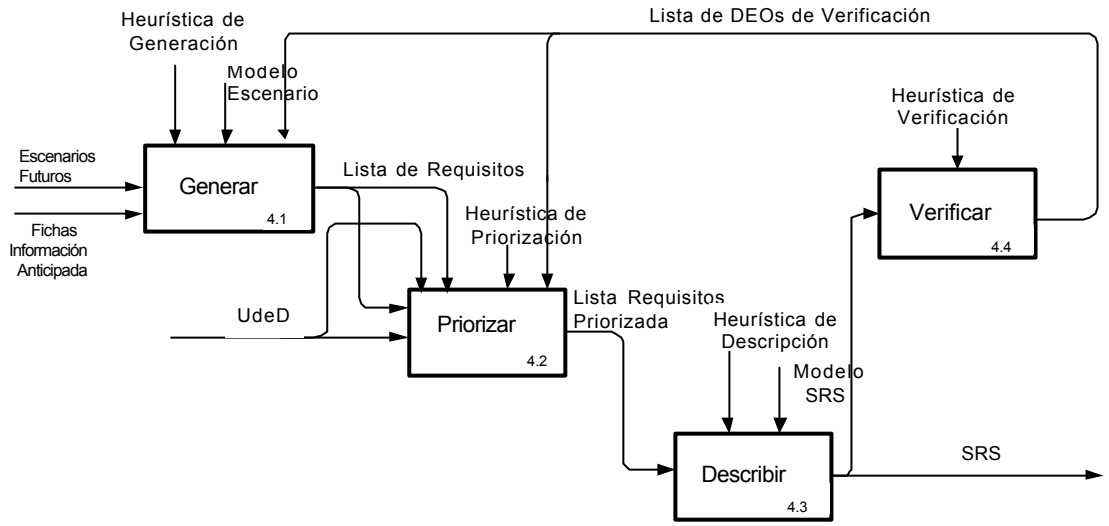


Figura 8-1. SADT del proceso de explicitar Requisitos de Software

(1) GENERAR

Esta actividad tiene por objetivo generar una lista de requisitos únicos del sistema de software, lo más completa posible. Incluye extraer los requisitos de cada EF, eliminar requisitos repetidos de la lista, y eventualmente obtener RNF de las Fichas de Información Anticipada que no fueron vinculados a ningún EF.

La sub-actividad extraer requisitos de cada escenario futuro puede considerar requisitos en dos niveles de detalle, según los componentes utilizados: requisitos de alto nivel o funcionalidades y requisitos detallados.

Requisitos de alto nivel se obtienen de los objetivos de cada EF, éstos pueden representar las grandes funcionalidades del sistema, siempre y cuando el actor sistema de software tenga una participación importante en los episodios del escenario. Este tipo de requisito puede descartarse por ser

redundantes respecto a los requisitos detallados, ya que los primeros se descomponen en los detallados. Por otro lado, su inclusión puede servir para establecer dependencias jerárquicas entre los requisitos, y en consecuencia establecer dependencias de cercanía entre los requisitos detallados. Esto es debido a la relación jerárquica entre los escenarios (debido al proceso de integración), que conlleva a una relación jerárquica entre los objetivos de escenarios, la cual se traslada a dependencias jerárquicas entre requisitos de alto nivel y requisitos detallados. Por otro lado, este tipo de relaciones puede establecerse mediante otros mecanismos, como la pertenencia al mismo EF (ver sección 8.3).

Requisitos detallados se obtienen de otros componentes e ítems del escenario, principalmente de los episodios donde el sistema de software actúa, éstos serán del tipo RF. Estos episodios no deben ser a su vez sub-escenarios. Además, pueden extraerse RF de las excepciones, donde la solución es una sentencia donde actúa el sistema de software, no siendo la solución un escenario excepción. En algunos casos las condiciones de los episodios condicionales pueden involucrar datos, estados de datos o acciones, y por lo tanto, pueden obtenerse RF. Esto mismo ocurre en la causa de excepciones.

Cabe aclarar respecto a las condiciones que los datos o estados de datos serán aquellos que el sistema deberá administrar, y en el caso de una condición que involucra una acción debe participar en ella el sistema. Se ha comprobado a través del estudio de varios casos que una condición que involucra una acción implica que se ha abreviado el discurso. Es decir:

Si acción1 **entonces** acción2 ⇒ acción1
Si verdadera **entonces** acción2

Ejemplo:

Si el sistema verifica que el equipo no tiene número de pedido **entonces** ASIGNAR NÚMERO DE PO

⇒

El sistema verifica que el equipo tiene número de pedido.
Si no tiene **entonces** ASIGNAR NÚMERO DE PO.

[82]

En el caso que la causa de una excepción involucre un RF, con frecuencia, significa que el sistema debe realizar algún tipo de verificación. Por ejemplo:

[82] Ejemplo extraído del caso "Sistema de Administración de Equipos" desarrollado para una empresa argentina por la Lic. Gladys Kaplan en el 2002.

Ejemplo 1:
Causa de excepción: El sistema no reconoce el número de formulario.
RF: El sistema debe verificar la existencia del número de formulario.

Ejemplo 2:
Causa de excepción: El pasaporte emitido no es retirado antes de los 60 días.
RF: El sistema debe verificar que la fecha de emisión del pasaporte no supere 60 días de la fecha actual cuando el solicitante pase a retirar el pasaporte..

[83]

Se pueden extraer RNF básicamente de las restricciones de los episodios donde participa el sistema de software, adicionalmente pueden encontrarse en la causa de excepciones. También a partir de los recursos involucrados, pueden inferirse algunos requisitos de calidad. Si el recurso es un símbolo tipo Objeto del LEL, puede buscarse en la noción del mismo alguna característica particular que pueda indicar la necesidad de un RNF. Por ejemplo:

Ejemplo 1:
Recurso: Medio de comunicación.
RNF: El sistema debe poder conectarse por algún medio de comunicación (e-mail) con todos los convocados a una reunión.

Ejemplo 2:
Recurso: archivo de presentación.
Noción del símbolo en el LEL: es un archivo plano que contiene todos los préstamos otorgados en un período de presentación. Cada pagador tiene un formato especial del archivo.
RNF: El sistema debe reconocer las distintas estructuras del archivo de presentación por pagador.

[84]

A continuación se presentan dos ejemplos de causa de excepción que implica un RNF:

[83] Ambos ejemplos extraídos del caso "Sistema Nacional para la Obtención de Pasaporte" desarrollado por profesores para la Universidad de Belgrano en 1996.
[84] El ejemplo 1 fue extraído del caso "Sistema de Agenda de Reuniones" desarrollado en 1997 durante el proyecto de investigación Uso de Escenarios en el Desarrollo de Software en la Universidad de Belgrano, y el ejemplo 2 fue extraído del caso "Sistema de Administración de Préstamos" desarrollado por alumnos del último de la carrera en la UNLaM en el 2006.

Ejemplo 1:
Causa de excepción: El sistema no se puede conectar al servidor donde se encuentran las estadísticas.
RNF: El sistema debe poder conectarse al servidor de estadísticas.

Ejemplo 2:
Causa de excepción: Problemas en el servicio de internet.
RNF: El sistema debe poder accederse desde internet.

[85]

Esto no significa que cada uno de estos componentes o ítems del escenario sea siempre un requisito, es el ingeniero de requisitos quien lo determina.

La Figura 8-2 muestra un ejemplo de lista de requisitos extraída de un solo escenario futuro, donde se aprecian cuatro RF y un RNF. En la Tabla 8-1 se resumen los posibles tipos de requisitos a extraer de los EF, dando ejemplos de cada uno de ellos. Se puede observar en los ejemplos de esta Tabla que existen palabras o frases subrayadas, ello representa vínculos a símbolos del LEL. En la Tabla 8-2 se identifica el modo de extracción de requisitos desde los componentes o ítems de un escenario.

[85] El ejemplo 1 fue extraído del caso "Sistema de Clasificados" para un diario argentino desarrollado por alumnos de 4º año de la UTN en el 2005, y el ejemplo 2 fue extraído del caso "Sistema de Administración de Préstamos" (ver nota al pie anterior).

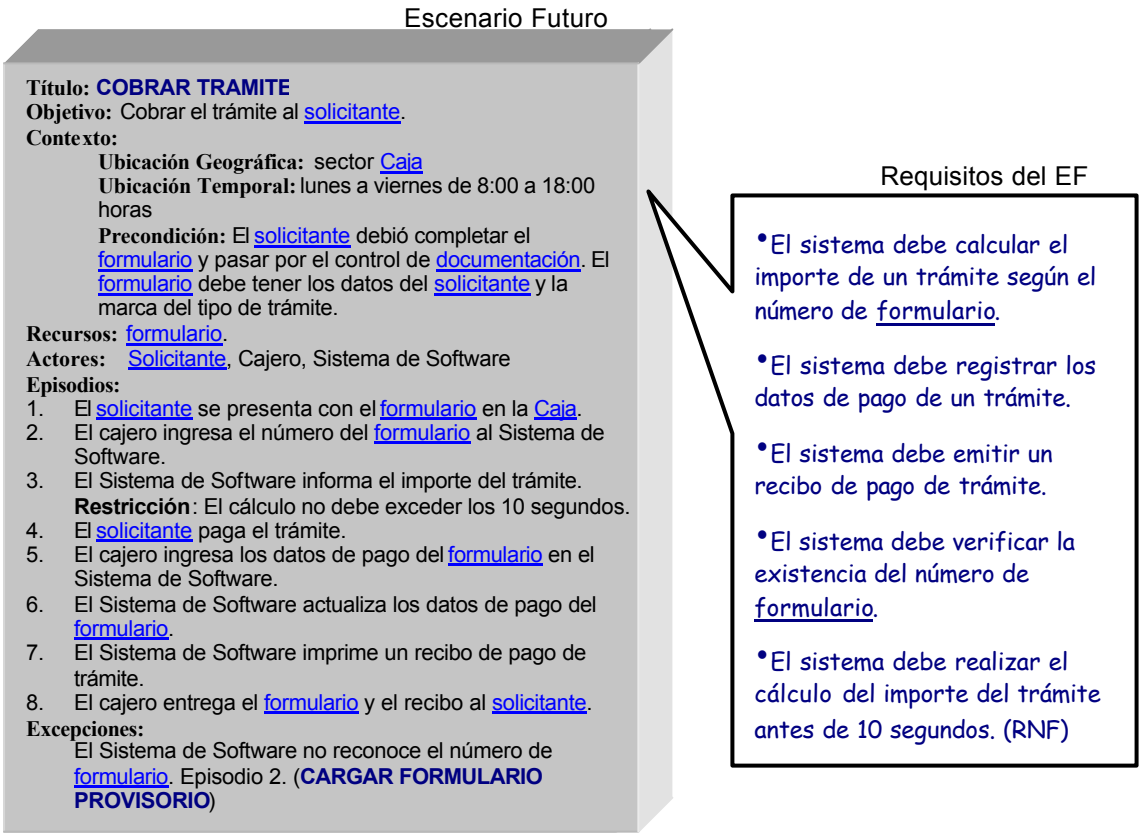


Figura 8-2. Ejemplo de lista de requisitos de un escenario futuro[86]

[86] Este ejemplo fue extraído del caso "Sistema Nacional para la Obtención de Pasaporte".

Nivel de Detalle	Tipo	Componente / Item	Ejemplo
Alto	*Funcionalidad*	*Objetivo* de los escenarios futuros	Cobrar el trámite al solicitante. Cargar el formulario provisorio.
Detallado	*Requisito Funcional*	*Episodios* donde participa el Actor Sistema de Software	El Sistema de Software imprime un recibo de pago del trámite.
		Solución de Excepción donde participa el Actor Sistema de Software	El Sistema envía mensaje sobre el límite de fecha para la incineración de un pasaporte.
		Condiciones donde en el episodio participa el Actor Sistema de Software	**Si** <<el tipo de trámite es pasaporte original>> **entonces** el Sistema … **Si** <<es el segundo chequeo del equipo>> **entonces** el Sistema … **Si** <<el Sistema verifica que el equipo no tiene asignado número de pedido>> **entonces** el Sistema …
		Causa de Excepciones	El Sistema no reconoce el número de formulario. Algún dato del convocado es inválido o nulo. El pasaporte emitido no es retirado antes de los 60 días. Administración modifica en el sistema algún dato cargado por Comercio Exterior.
	Requisito No Funcional	*Restricción* donde en el episodio participa el Actor Sistema de Software	El Sistema debe realizar el cálculo del importe del trámite antes de 10 segundos.
		Causa de Excepciones	Existen problemas en el servicio de internet. El Sistema no se puede conectar al servidor donde se encuentran las estadísticas.
		Recursos	e-mail archivo de presentación

Tabla 8-1. Tipos de requisitos extraídos de escenarios futuros

Se debe tener presente que dependiendo del componente o ítem puede extraerse directamente un requisito, puede evaluarse si es un requisito o puede inferirse un requisito. Detallando para cada caso expuesto en la Tabla 8-1, se observa en la Tabla 8-2 los tipos de relación para extraer requisitos de los componentes o ítems del escenario.

Componente / Ítem	Relación con requisito
Objetivo	PUEDE SER
Episodios	ES
Solución de Excepciones	ES
Condiciones	PUEDE INFERIRSE o PUEDE SER (caso acción)
Causa de Excepciones	PUEDE INFERIRSE
Restricción	PUEDE SER
Recursos	PUEDE INFERIRSE

Tabla 8-2. Tipos de relación entre componente y requisito

Una vez extraídos los requisitos detallados de cada EF, debe controlarse la existencia de redundancia de los mismos. Es muy frecuente que un requisito esté presente en más de un escenario. Por ejemplo el requisito *"El sistema debe permitir la selección de un cliente por apellido, por tipo o por localidad"* puede presentarse en más de un escenario donde debe realizarse alguna tarea vinculada a un cliente. Otro ejemplo, *"El tiempo de respuesta en la consulta del comprobante no debe exceder los 30 segundos"*, puede estar vinculado a todas las consultas de diversos comprobantes, y puede generalizarse para todos ellos. Luego a partir, de esta sub-actividad, se obtiene una lista de requisitos únicos.

Finalmente, se revisa cada Ficha de Información Anticipada no utilizada en la etapa anterior y que esté marcada como RNF, para extraer aquellos RNF que no pudieron aparease con ningún EF, por ser más abarcadores (aunque pudiesen ligarse a escenarios futuros integradores) o estar fuera del ámbito de los procesos del negocio. Por ejemplo, *"El sistema debe trabajar bajo una red que soporte un máximo de 100 usuarios trabajando simultáneamente sobre el sistema"* o *"El sistema debe mantener la privacidad de los datos monetarios mediante el método de encriptado XYZ"*. De esta manera se completa la lista de requisitos previamente confeccionada.

Estos RNF deben validarse con los clientes y usuarios pues fueron extraídos de Fichas de Información Anticipada. Los ingenieros de requisitos

deben verificar que no estén en conflicto con otros requisitos, si ello ocurre deberán negociarse con los clientes y usuarios.

(2) PRIORIZAR

Los clientes y usuarios dan prioridades a los requisitos con apoyo de los ingenieros de requisitos (ver técnicas de priorización de requisitos en sub-sección 2.2.6). Se realizan reuniones para determinar la importancia relativa que tiene un requisito para los clientes y usuarios, y también para organizar aquellos requisitos que deben implementarse inicialmente frente a aquellos que pueden postergarse.

Para que esta actividad pueda realizarse correctamente, debe primero establecerse la dependencia entre los requisitos. Estas relaciones pueden ser de los siguientes tipos:

- ✓ Relaciones entre RF dada por la vinculación al mismo EF.
- ✓ Relaciones entre RF y RNF dada por la vinculación entre episodio y restricción en el EF.
- ✓ Relaciones entre un RNF aplicado a un episodio que es escenario y los RF extraídos del sub-escenario.
- ✓ Relaciones entre requisitos obtenidos de excepciones y requisitos que pueden dar origen a la excepción.
- ✓ Relaciones entre requisitos establecidas semánticamente.

Se debe tener en cuenta que no sólo se le da prioridad a los requisitos sino también puede establecerse su criticidad, factibilidad y riesgo, que son algunos de los atributos que presentan los requisitos (ver sub-sección 2.6.3). Para dar prioridades se debe considerar, además de la dependencia de los requisitos, la diversidad de intereses de los clientes y usuarios, las limitaciones de recursos, las necesidades del negocio y las imposiciones del mercado, entre otros factores.

Esta tarea puede realizarse en la etapa anterior dando prioridad a los EF, dado que puede ser más fácil que priorizar cada requisito individualmente, ya que muchos de ellos están interrelacionados y deben considerarse sus prioridades en simultáneo.

(3) DESCRIBIR

Los requisitos se describen según el formato de SRS impuesto por el cliente o adoptado por el equipo de desarrollo. El propósito principal es organizar los requisitos de manera tal de obtener el máximo de legibilidad. Existen estándares como [IEEE Std 830-1998] que presentan 8 formas diferentes de presentar bs requisitos según las características del sistema a construir.

Deben mantenerse en el SRS vínculos al LEL, como una forma de reducir la ambigüedad del documento escrito en LN. Adicionalmente se mantienen vínculos a los escenarios futuros para mantener la pre-trazabilidad de los requisitos. Debe además considerarse que muchos requisitos están relacionados con otros, en muchos casos se trata de RNF con RF. El mantenimiento de estos vínculos permite la inter-trazabilidad (ver sub-sección 2.2.8).

Recomendaciones para describir cada requisito (algunas extraídas de [Hull 05]):

- ✓ Redactar utilizando la forma *"El sistema debe …"*, con lo cual se identifica claramente la capacidad o la condición que el sistema brindará o acatará.
- ✓ No incluir más de un requisito en cada sentencia.
- ✓ Evitar el uso de la frase "de ser necesario", pues debe quedar claro bajo qué condiciones opera el requisito.
- ✓ Evitar palabras ambiguas como "generalmente", "frecuentemente", "rápido", "flexible", "amigable", "preferible".
- ✓ Evitar deseos ilimitados en las capacidades del sistema, por ejemplo "bajo cualquier plataforma", "independiente del motor de base de datos", "100% confiable", "sin fallas".
- ✓ Para RNF, incluir valores de medición. Se puede describir el valor deseado y el valor máximo o mínimo aceptable.

Si se considera el formato presentado por [IEEE Std 830-1998], entonces gran parte de sus secciones (ver sub-sección 2.6.5) puede completarse con la información obtenida, como se muestra a continuación. La sección 1.2 del SRS "Alcance del Proyecto" incluye el nombre del proyecto que puede obtenerse del título del escenario futuro integrador de nivel 0, y objetivo del producto que se obtiene del documento de objetivos del sistema o de los objetivos de los escenarios integradores. En la sección 1.3 del SRS "Definiciones, acrónimos y abreviaturas" puede hacerse referencia al LEL. En la sección 1.4 "Referencias" se indicarán referencias al conjunto de EF y al LEL. La sección 2.2 "Funciones del producto" se puede completar con los objetivos de los EF con el sistema como actor importante. En la sección 2.3 "Características de los usuarios" se puede completar con los actores de los EF y vincularlos con los respectivos símbolos tipo Sujeto del LEL. La sección 3 "Requisitos Específicos" incluye todos los requisitos extraídos de los EF, donde podrán organizarse por su tipo, por el EF origen, por prioridad, por Actor del EF.

(4) VERIFICAR

La actividad de verificación debe comprobar la correcta generación del SRS, determinando si la extracción de requisitos desde los EF fue realizada exhaustivamente, si la priorización es consistente y si el SRS contempla todos los requisitos explicitados y priorizados. Dado que la verificación de los otros modelos de SDRES se realiza utilizando la técnica de inspección, es

aconsejable aplicarla también a este documento, ya que los ingenieros de requisitos estarán instruidos sobre la técnica. Se genera a partir de esta actividad una lista de DEOs para corregir el SRS.

Los aspectos principales que cubre la verificación son:

- ⇨ Verificar la sintaxis:
  - ✓ Detectar el uso de más de un verbo en la descripción del requisito.
  - ✓ Detectar el uso de adjetivos subjetivos.
  - ✓ Detectar el uso de frases ambiguas o subjetivas (cuyo significado depende del interlocutor).

- ⇨ Verificar contra los componentes del EF:
  - ✓ Controlar que todo requisito proviene de un componente del EF: objetivo, episodio, restricción, causa de excepción, solución de excepción, recurso o condición de episodio.
  - ✓ Controlar por excepción que un RNF proviene de una Ficha de Información Anticipada, marcado en Tipo de Información como RNF y para ser incluido en el SRS.

- ⇨ Verificar dependencias entre requisitos:
  - ✓ Detectar relaciones no establecidas según sus orígenes en EF.

- ⇨ Verificar prioridades:
  - ✓ Controlar que la prioridad asignada a un requisito esté dentro del rango establecido.
  - ✓ Controlar que la prioridad del requisito sea consistente según su dependencia con otros requisitos.

- ⇨ Verificar referencias al LEL:
  - ✓ Controlar que todo símbolo del LEL es identificado al mencionarse en un requisito.
  - ✓ Controlar el uso correcto de símbolos del LEL.

Con la lista de DEOs generada se vuelve a las actividades Generar la lista y Priorizar los requisitos según el tipo de defectos encontrados.

8.3. Gestión de requisitos

La gestión de requisitos (ver sub-sección 2.2.7) se ve facilitada al tener individualizados los requisitos en un documento, de manera que al solicitar la incorporación de nuevos requisitos o la actualización de requisitos establecidos se pueda analizar que otros requisitos se verán afectados y que modelos o componentes de software deberán actualizarse, en función de la etapa de desarrollo. Para ello se debe establecer una adecuada trazabilidad entre los documentos, modelos y otros artefactos generados durante el desarrollo del software, como así también definir un sistema de versionado de los mismos.

Se analiza a continuación los vínculos entre modelos, algunos propios de la naturaleza misma de los modelos y otros específicos que hacen a su trazabilidad.

Los vínculos provenientes de los modelos que administra el proceso pueden ser internos o externos a cada modelo (señalados en la Figura 4-1 del capítulo 4) dentro del CORB:

- Vínculos internos:
  - Dentro del LEL por referencias a símbolos (indicados en la Figura 4-1 con flecha completa)
  - Dentro del conjunto de EA por relación jerárquica (indicado en la Figura 4-1 con flecha punteada)
  - Dentro del conjunto de EF por relación jerárquica
  - Dentro del conjunto de EA por relación de excepción
  - Dentro del conjunto de EF por relación de excepción (indicado en la Figura 4-1 con flecha punteada)

- Vínculos externos (indicados en la Figura 4-1 con flecha guionada):
  - De EA a LEL por referencia a símbolos
  - De EF a LEL por referencia a símbolos
  - De SRS a LEL por referencia a símbolos

Adicionalmente a los vínculos semánticos naturales de los modelos de SDRES, es necesario mantener otros vínculos que permitan el rastreo de los requisitos a lo largo de éstos modelos con fines de vincular la forma de obtención de los mismos siguiendo la estrategia de construcción o para establecer dependencias entre requisitos. Estos vínculos (indicados con flechas llenas en la Figura 8-3) se establecen dentro del CORB, en algunos casos podrán generarse automáticamente por el proceso de creación correspondiente (por ejemplo al derivar un EA de un símbolo del LEL o al extraer un requisito de un EF) y en otros deberá usarse un método manual (por ejemplo al relacionar un requisito con otro u otros requisitos). Estos vínculos de rastreo son bi-direccionales y establecen relaciones N a M entre los modelos.

Para cada requisito descripto en el SRS, se indican nexos a los EF que le dieron origen y nexos a otros requisitos, que implican relaciones de dependencia. Los primeros vínculos permiten una pre-trazabilidad al origen del requisito y los segundos muestran la inter-trazabilidad. Esta última es de gran utilidad dado que para cualquier cambio en un requisito, es necesario conocer si afecta a otros requisitos. La trazabilidad hacia atrás permite determinar cuales EF serán alterados frente a un cambio y por lo tanto, cómo repercute en el resto de los escenarios futuros y permitir justificar dicho cambio. Adicionalmente, este vínculo a los EF le agrega detalle al requisito.

En la Figura 8-3 se observan en flechas llenas las trazas dentro del CORB referidas a la pre-trazabilidad e inter-trazabilidad (no se han incluidos los vínculos naturales de los modelos), y las trazas hacia otros artefactos externos al CORB se dibujan con flechas punteadas. También se indica en esta figura el origen de la traza; por ejemplo la relación entre un Requisito y uno o varios EF

se obtiene mediante el proceso de extracción de requisitos desde EF; otro ejemplo es la relación entre un EA y un símbolo del LEL que se establece al derivar del LEL impactos de símbolos tipo sujeto generando un EA. Las trazas son bi-direccionales, es decir de un EF se puede establecer los requisitos que contiene en el SRS y de un requisito identificar los EF que le dieron origen. En el caso de las Fuentes de Información se han presentado los vínculos como unidireccionales, pues no se justifica una búsqueda desde la fuente a un modelo, sino la inversa, desde un modelo poder determinar el origen de la información. Se debe tener presente que la representación de las trazas no altera los modelos.

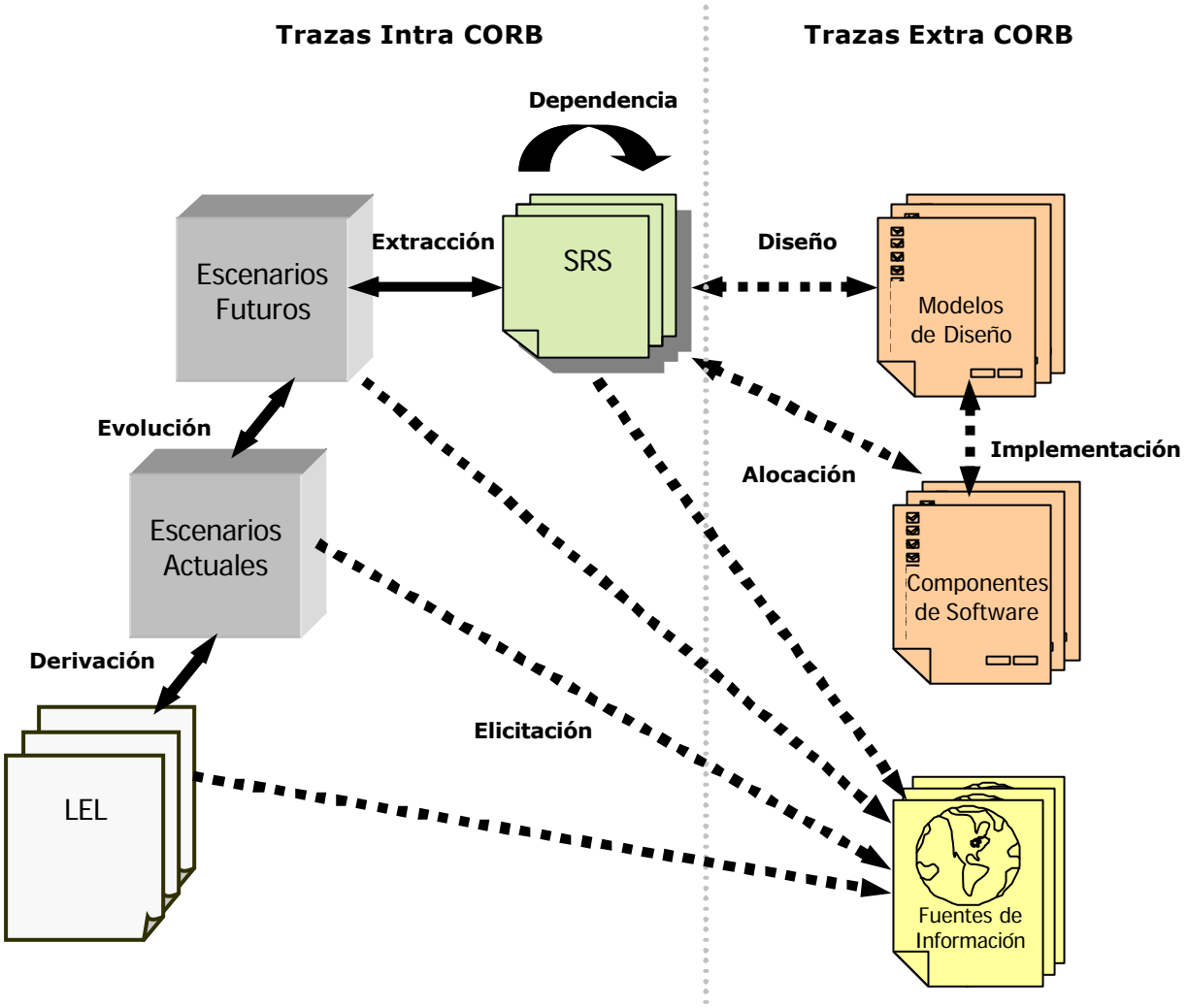


Figura 8-3. Trazas semánticas Intra CORB y Extra CORB

Las relaciones de dependencia entre requisitos se establecen durante la actividad Priorizar, porque no sólo hacen a la trazabilidad de los requisitos sino también es fundamental al momento de darles prioridades.

Por otro lado, se requieren vínculos de versionado (ver sección 3.6) para permitir una búsqueda o recorrido correcto entre los elementos de un modelo y hacia otros modelos, que complementan los vínculos naturales y los vínculos de rastreo. Luego, para una dada versión del SRS debe establecerse la

relación correcta con una versión del conjunto de EF y con una versión del LEL. Como ya se ha mencionado en la sección 3.6, un modelo de trazas por apilamiento para el versionado será lo más adecuado por su facilidad de mantenimiento y por su independencia respecto a la administración de los modelos de SDRES.

8.4. Observaciones

Existen en la literatura muchas propuestas sobre los escenarios y casos de uso y, por otro lado, mucho se dice sobre los requisitos de software, pero poco se trata sobre extraer requisitos de los escenarios y cómo redactar un SRS no ambiguo. En general, se puede decir que muchas estrategias parten de un documento o lista de requisitos, cuyo origen no es explicado, y a partir del cual se resuelven conflictos, se dan prioridades, se gestionan los cambios en los requisitos y se alocan los mismos. Es decir, existen muchas actividades de gran importancia en el desarrollo del software que están motivadas a partir de un conjunto de requisitos cuya elicitación y definición no están expuestas ni justificadas, y donde dicho conjunto pareciera surgir espontáneamente en el proceso de desarrollo.

Otro punto importante es que los requisitos extraídos de los EF no presentan conflictos entre sí. Esto se debe a que ya en el proceso de construcción de los escenarios futuros los conflictos se identifican con técnicas de V&V y se solucionan a través de la negociación.

Existen diversas estrategias para la identificación y análisis de conflictos (ver sub-sección 2.2.6, Nota al pie 18). Con lo cual, el tema de requisitos contradictorios está ampliamente difundido en la literatura, pero en general poco se menciona sobre la forma en que fueron elicitados y definidos los requisitos; es probable que los conflictos entre requisitos surjan cuando éstos han sido levantados en una modalidad ad-hoc sin ningún procedimiento que los respalde, y por lo tanto no han soportado la consistencia implícita que se produce por la construcción de los escenarios futuros con heurísticas que obligan a integrar las situaciones imaginadas y, por lo tanto, las contradicciones no tienen ninguna posibilidad de prosperar en la lista de requisitos.

En principio esta tesis genera una lista de requisitos clasificándolos sólo en RF y RNF, pero los cuales están validados y acordados por los clientes y usuarios. A partir de ella, puede redactarse un documento de definición de requisitos organizando los requisitos de acuerdo al estándar utilizado, ya sea éste elegido por la organización cliente o por los desarrolladores. La ambigüedad del SRS se reduce pues estará redactado utilizando los términos empleados en el UdeD y con trazas a los EF que les dieron origen.

Cabe mencionar que en el Proceso Unificado [Jacobson 99] se reemplaza la SRS por un conjunto de modelos: casos de uso, modelo del dominio o del negocio y lista de requisitos adicionales. Es decir, no pone en evidencia cada requisito, y descarta de llano la creación de un documento.

8.5. Resumen

Un aspecto a destacar es que la calidad de los requisitos obtenidos de los EF está garantizada pues ellos han sido previamente verificados, validados y acordados, y además estos requisitos no presentan conflictos.

Muchos requisitos del software se pueden obtener en forma inmediata de los EF meramente redactando de manera diferente las acciones en las que participa el sistema de software. Pero esto no se puede extender más allá de cierto límite ya que existen en los EF otros requisitos que no son tan evidentes ni tan fáciles de explicitar. Es precisamente a esto a lo que está abocado este capítulo, es decir a describir las diferentes actividades que se deben realizar para explicitar lo más posible los requisitos presentes en los EF.

En la práctica, debería decirse que una buena definición de requisitos de software debiera carecer de RNF o minimizar su cantidad, ya que en su gran mayoría pueden convertirse en comportamientos del sistema.

En la Tabla 8-3 se exponen estadísticas sobre casos donde se aplicó el proceso de extracción de requisitos a partir de EF. Se ha contabilizado el total de requisitos extraídos de los EF considerando: episodios, condiciones de episodios, restricciones, soluciones simples de excepciones y causas de excepciones, que representasen algún tipo de requisito. En resumen, se obtuvieron 1271 requisitos en 15 casos de estudio.

Además, en esta tabla se contrasta para varios de estos casos la cantidad de requisitos obtenidos de los EF en una modalidad ad-hoc frente a la aplicación del proceso expuesto en este capítulo, donde se puede observar que en algunos casos se ha llegado a duplicar la cantidad de requisitos usando el proceso presentado.

Año	Caso de Estudio	Institución	Autores	Cantidad total de Escenarios	Cantidad de Episodios	Cantidad Requisitos Ad-hoc	Cantidad Requisitos Derivados
2002	Sistema de Administración de Cajeros automáticos	Empresa	1 Ingeniero de Software	38	195	--	167
2004	Planificación y Seguimiento en un Laboratorio Farmacéutico	UTN-FRBA	3 Alumnos de grado	20	142	--	78
2005	Administración del Almacén de MP y artículos de Encuadernación	UNLaM	3 Alumnos de grado	17	86	42	67
2005	Seguimiento de Service de equipos para radiocomunicación	UNLaM	2 Alumnos de grado	12	55	32	117
2005	Sistema de Clasificados de un diario	UTN-FRBA	2 Alumnos de grado	23	94	22	27
2005	Gestión de Producción de Químicos	UTN-FRBA	2 Alumnos de grado	29	171	45	79
2005	Administración de Soportes con contenidos de programación por cable	UTN-FRBA	2 Alumnos de grado	20	81	19	24
2005	Servicio de Ecodoppler y Ecocardiografía	UTN-FRBA	3 Alumnos de grado	22	103	49	54
2005	Seguimiento de Producción de Cajas de Cartón – Versión 1	UNLaM	2 Alumnos de grado	10	58	--	52
2006	Seguimiento de Producción de Cajas de Cartón – Versión 2	UNCPBA	1 Alumno de grado	8	43	--	286
2006	Administración de Préstamos	UNLaM	3 Alumnos de grado	20	103	21	50
2007	Gestión de convenios universitarios	UNLaM	2 Alumnos de grado	22	91	--	45
2007	Servicio de Presupuesto y Seguimiento de Pedidos	UNLaM	2 Alumnos de grado	38	171	37	84
2007	Sistema de Agenda de Reuniones – Versión 1	UNLaM	1 Investigador	17	90	--	22
2007	Sistema de Agenda de Reuniones – Versión 2	UNLaM	1 Investigador	24	112	81	119

Tabla 8-3. Datos de casos de estudio sobre Requisitos derivados de EF

En la Figura 8-4 se muestra que la primera actividad Generar la lista de requisitos es netamente una actividad de modelado, mientras que Priorizar los requisitos involucra elicitación, negociación y modelado. Luego Describir el SRS implica modelar y Verificar el SRS implica una actividad de análisis.

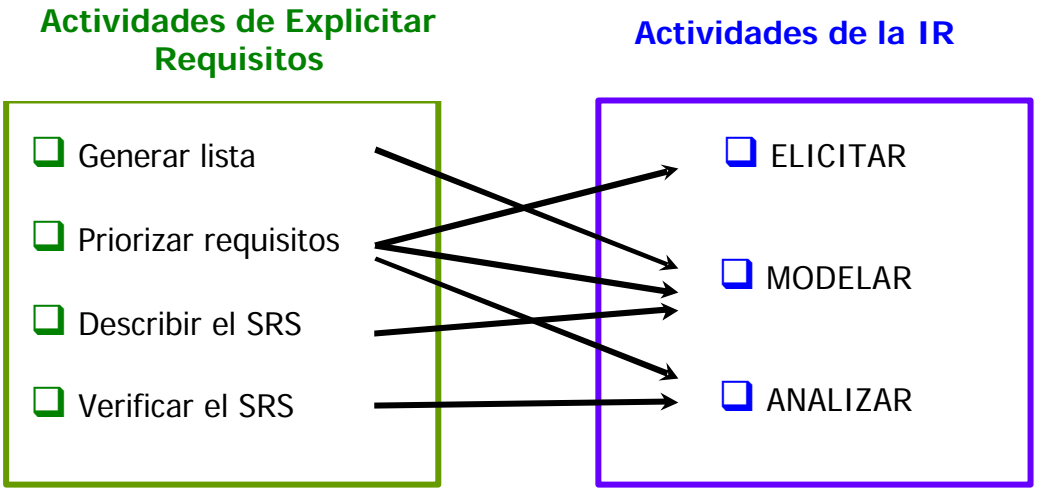


Figura 8-4. Relación entre actividades de IR y actividades de explicitar requisitos

La gestión en la IR involucra el mantenimiento de versiones de SRS y la trazabilidad de los requisitos hacia los EF donde están empotrados, y trazas hacia modelos externos al CORB, tales como modelos de diseño, modelos de implementación y componentes de software.

Parte del material incluido en este capítulo ha sido publicado en [Leite 04b].

Capítulo 9
Conclusiones

«If a system is to evolve and grow as its users' requirements and skills evolve, it must be founded on well-engineered software abstractions»

M.B. Rosson, "Integrating Development of Task and Object Models", ACM, 1999

9.1. Corolario

Se ha desarrollado un método que abarca todas las actividades de la IR, que está dirigido por modelos que facilitan la elicitación y está orientado al cliente mejorando la comunicación y facilitando la validación de los requisitos.

SDRES apunta a obtener una mejor comprensión del problema con el fin de proponer las mejores soluciones negociadas. Establece un puente entre las necesidades de los clientes y usuarios, y la solución de los diseñadores. La estrategia puede obtener las mejores respuestas a preguntas tales como:

- ⇨ ¿Cuáles son las necesidades?
- ⇨ ¿Son realmente necesidades?
- ⇨ ¿Por qué son requeridas?
- ⇨ ¿Cuáles son los problemas actuales?
- ⇨ ¿Cómo pueden solucionarse los problemas?
- ⇨ ¿Cómo pueden tratarse o cubrirse las necesidades?

Se han propuesto heurísticas para la elicitación, el modelado y el análisis de los requisitos, presentando distintas alternativas para transitar la estrategia de acuerdo al problema a resolver.

Se han refinado y ampliado las heurísticas para la construcción del LEL, presentando nuevos problemas y soluciones, tales como el tratamiento de homónimos, las relaciones de sinonimia e hiperonimia (tipo y super-tipo), uso del género en la noción, consideración del mecanismo de creación de los términos y la evolución del LEL. Se ha enfatizado la búsqueda de fuentes de información y la perspectiva de la vigencia de información que ellas trasuntan.

Respecto a los escenarios, se ha actualizado el modelo de escenarios en base a la experiencia obtenida en la variedad y cantidad de conjuntos de escenarios construidos, tanto en casos de estudio como casos reales.

Se han refinado las heurísticas para la construcción de escenarios actuales, principalmente la realización de operaciones y la creación de escenarios integradores. También se ha mejorado la heurística de descripción y se ha dado una guía para el uso de roles en caso de actores alternativos en un escenario. Se han comparado los posibles enfoques top-down y bottom-

down frente a la modalidad middle-out presentada en la tesis para la construcción de escenarios actuales.

Se ha comprobado a través de su uso en varios casos de estudio que las inspecciones son una técnica de verificación de gran efectividad aplicada a documentos de requisitos.

Se han presentado heurísticas para la evolución de escenarios actuales a escenarios futuros bajo el marco de los objetivos del sistema de software. Se han comparado éstas con aquellas propuestas por otros autores que manejan objetivos y escenarios. Se ha enfatizado en la distinción entre los objetivos de los procesos del negocio y los objetivos del sistema, considerando que los escenarios introducidos en esta tesis representan situaciones bajo un contexto organizacional. Se ha incentivado la elicitación de RNF que puedan asociarse al comportamiento que involucran los escenarios.

Se han expuesto heurísticas para poner en evidencia los requisitos del software, facilitando su trazabilidad y permitiendo la creación de un SRS.

Se ha descripto la evolución de los modelos, tipificando las causas de dicha evolución y cómo gestionarla. Se ha justificado el mantenimiento de los modelos que componen SDRES y se ha tratado su versionado. Así mismo, se ha expuesto cómo aplicar SDRES y cómo adaptarlo según las características del problema propiamente dicho y de la organización cliente.

9.2. Consideraciones finales

La propuesta de esta tesis para la construcción de escenarios actuales, una estrategia middle-out, intenta ser una contribución a la diversidad de propuestas existentes. Esta creencia se basa en la experiencia previa de haber utilizado una estrategia top-down. La diferenciación entre escenarios actuales y escenarios futuros clarifica el proceso de la IR. En tal sentido, muchas estrategias hablan sólo de los "to-be scenarios"[176] dejando de lado el conocimiento actual del UdeD. Esto induce a crear escenarios futuros menos anclados en la cultura del UdeD que lo que se logra creando previamente los escenarios actuales.

En muchos casos, existe un alto grado de inculturización[177] de sistemas en las organizaciones o transculturización[178] de organizaciones por culpa de los sistemas. No hay que aliarse con el preconcepto que la transculturización es per se mala, pero sí que se debe tener conciencia acerca de si se va a producir o no.

[176] En la literatura, se menciona a los "as-is scenarios" que representarían los escenarios actuales, usando el término de "to-be scenarios" para los escenarios futuros.
[177] Inculturización: agregar algo a una cierta cultura en forma natural y coherente, respetando sus estructuras básicas y sólo extendiendo las mismas.
[178] Transculturización: agregar algo a una cierta cultura en forma abrupta y antagónica con sus estructuras básicas.

La Tabla 9-1 muestra cómo distintos métodos encaran las actividades de la IR, donde se observa que la mayoría de los métodos hacen énfasis en el modelado y pocos en la elicitación, siendo la gestión uno de los tópicos menos tratados.

Método / Actividad	Análisis Estructurado	SCR	RUP	XP	SDRES
Elicitación	☑	---	☑	☑☑	☑☑
Modelado	☑☑☑	☑☑☑	☑☑☑	☑	☑☑☑
Análisis	---	☑☑☑	☑☑	☑	☑☑☑
Gestión	---	---	☑☑	☑☑	☑☑

☑ bajo ☑☑ medio ☑☑☑ alto --- nulo

Tabla 9-1. Énfasis de diversas estrategias en las actividades de la IR

Tratar con la completitud en estrategias semi formales es extremadamente difícil. Esto es también verdadero en la estrategia descripta. Sin embargo se han realizado esfuerzos significativos para ser exhaustivo a lo largo de todo el proceso. Los puntos clave sobre esto son:

i) El LEL es consistentemente incrementado y mejorado a lo largo del proceso.
ii) La integración de escenarios actuales puede sugerir la existencia de gaps semánticos entre escenarios. Lo mismo ocurre para la integración de escenarios futuros.
iii) La construcción de escenarios futuros combinando los objetivos del sistema de software con escenarios actuales amplía el espectro de la elicitación.
iv) Verificar y validar los modelos producidos indica un proceso orientado a la calidad.

La propuesta original de Leite en [Leite 95] [Leite 97] y aún versiones posteriores en [Breitman 05] hacen referencia a un único conjunto de escenarios que evoluciona desde dos perspectivas. Una evolución se presenta por el avance en el proceso de desarrollo, considerando escenarios de especificación, escenarios de diseño y escenarios de implementación, y otra evolución referida al mantenimiento de dichos escenarios por cambios en los requisitos. Todos estos escenarios son portadores del mismo conjunto de requisitos pero expresados en distintas fases del desarrollo. Es decir, cada uno de estos tipos de escenarios tiene un uso diferente, mientras los primeros sirven para elicitar, comunicar y negociar requisitos con los usuarios, los escenarios de diseño son útiles a los diseñadores para encontrar objetos y el último tipo sirven para modelar los objetos de implementación. El primer tipo de evolución excede las actividades de la IR. La estrategia que se propone en esta tesis trata la evolución de los escenarios en la IR pero no propone la extensión

de dichos modelos en etapas posteriores de la IR, lo cual no indica la evolución de los mismos por cambios en los requisitos.

El CORB propuesto por Leite en [Leite 95] y aceptado en esta tesis con alguna variante, incluye los varios modelos de requisitos presentados en esta estrategia, orientados al cliente por ser escritos en LN y utilizando el léxico del cliente. Actualmente la propuesta de Breitman & Leite [Breitman 05] incluye en el requirements baseline escenarios de especificación, de diseño y de implementación, como también tarjetas CRC y modelo lógico. Es decir, no incluye sólo artefactos orientados al cliente.

Se debe enfatizar que la investigación en la IR enfrenta un problema de experimentación significativa [Ravid 00] [Kaindl 02], ya que se están desarrollando propuestas metodológicas para sistemas medianos, medianos-grandes y grandes pero los casos de estudio se realizan con problemas a veces visualizados como "problemas de juguete". Esto trae la grave consecuencia que el observador casual percibe en la propuesta una inmensa burocracia frente a un problema que definitivamente no lo justifica. Es tal vez el caso expuesto en el Apéndice D, "Sistema de Agenda de Reuniones", el cual fue elegido por su conocimiento ampliamente difundido y principalmente por no ser excesivamente extenso para su presentación completa. Cabe mencionar que la estrategia fue aplicada no sólo a casos de estudio de diverso tamaño y en distintas universidades, sino también en casos reales por diversos profesionales.

La autora de la tesis ha sido revisora de aproximadamente 150 casos de estudio siguiendo esta estrategia, realizados por alumnos de grado avanzados en las universidades UTN y UNLaM, aplicados en casos reales en distintas organizaciones argentinas. La estrategia ha sido promovida en cursos de postgrado asistidos por profesionales del país y extranjeros.

9.3. Trabajos Futuros

Se enuncia a continuación una serie de temas expuestos en esta tesis que se abren para investigaciones futuras.

Jerarquías en el LEL: esta tesis sólo intenta señalar estas relaciones descubiertas después de la creación de varios léxicos, dejando para una investigación posterior un análisis consciente y un tratamiento apropiado, especificando un proceso de detección.

Modelo de trazas: la tesis presentó un bosquejo de modelo de trazas, pero no se ha ahondado en el tema sobre el tratamiento de las trazas a modelos externos al CORB. El modelo de trazas es preliminar y deberá aplicarse en varios casos de estudio de diferente nivel de complejidad para poder obtener un modelo refinado que abarque todos los posibles rastreos de requisitos.

Diseño de arquitectura: otro tema abierto se refiere a la relación entre

los escenarios futuros y el diseño de la arquitectura. Si se parte de un conjunto pre-existente de posibles arquitecturas, los escenarios futuros pueden estar excesivamente influenciados por esta condición, considerando entonces un proceso orientado al reuso. Por otro lado, si el software es suficientemente ingenuo, entonces es muy probable que los escenarios futuros sean el punto de partida donde tener en cuenta el diseño.

Negociación: es también un tema de futura investigación cómo los escenarios futuros pueden impactar las negociaciones entre los ingenieros de requisitos, los diseñadores y los clientes. Esto es particularmente verdadero en el caso de sistemas que dependen de interfaces orientadas al usuario.

Elicitación: un tema a ampliar es considerar un modelo de fuentes de información, y cómo detectar eficientemente los usuarios claves y sus relaciones sociales en la organización.

Validación: es un problema abierto en el que podrían surgir nuevas contribuciones que mejoren la actividad, principalmente en la validación de escenarios futuros que requieren un esfuerzo extra por parte de los usuarios al tratar sobre un contexto no existente aún.

Apéndice A
Principales iniciativas en la Ingeniería de Requisitos

«Requirements analysis poses the problem of developing a mutual understanding of an artefact which does not exist»

A. Sutcliffe, "A Technique Combination Approach to Requirements Engineering", IEEE, 1997

A.1. Introducción

A pesar de ser la IR una disciplina reciente, cuyos comienzos datan de principios de la década del 90, existen abundantes evidencias de la dimensión que ha tomado en el desarrollo de sistemas de software. Esta evidencia se basa en la serie de estándares fijados para el documento de especificación de requisitos y para el proceso de producción de los requisitos, la variedad de productos comerciales que soportan de diversas formas las actividades involucradas en la IR, la profusión de congresos, revistas y libros específicos en esta área y las comunidades informáticas dedicadas al estudio de la informática.

A.2. Fijación de estándares para especificar requisitos

- **IEEE Std 830-1998**, IEEE Recommended Practice for Software Requirements Specifications, IEEE, New York, 1998.
- **IEEE Std P1233/D3**, IEEE Guide for Developing System Requirements for Specifications, IEEE, New York, 1995.
- **DOD-STD-2167A**, Defense System Software Development, US Department of Defense, Febrero 1988. Reemplazado en Diciembre de 1994 por MIL-STD-498 (el cual elimina la influencia del modelo de cascada).
- **NCC**, The STARTS Guide: A Guide to Methods and Software Tools for the Construction of Large Real-Time Systems, National Computing Centre, Manchester, 1987.
- **MIL-STD-498**, estándar requerido por el gobierno de Estados Unidos, Diciembre 1994, cancelado en Junio 1998. Este estándar implementa los procesos de desarrollo y documentación de ISO/IEC DIS 12207, e interpreta todas las cláusulas aplicables en MIL-Q-9858A (Quality Program Requirements) y en ISO 9001 (Quality Systems) para software. Establece un template para el Documento de Especificación de Requisitos y describe un proceso de desarrollo de software incremental e iterativo y prácticas para la evolución de requisitos.
- **ESA PSS-05-01**, Guide to the software engineering standards, desarrollado por European Space Agency Board for Software Standardisation and Control, Octubre 1991. En particular presenta dos documentos referidos a requisitos: **ESA PSS-05-02** "Guide to the User Requirements Definition Phase" y **ESA PSS-05-03** "Guide to the Software Requirements Definition

Phase".

- **DOD 5000.2**, estándar requerido por el gobierno de Estados Unidos, 2000. Este estándar reemplazó al MIL-STD-498, y también recomienda el uso de procesos incrementales e iterativos.

A.3. Estándares de calidad para procesos de software que han incluido estándares para el proceso de IR

- **SW-CMM**, Capability Maturity Model for Software, desarrollado por el Software Engineering Institute, Universidad de Carnegie Mellon, 1991. Este modelo en el nivel 2 repetible incluye a la gestión de IR como una área de proceso clave; [CMM 95] [Paulk 93].
- **CMMI**, Capability Maturity Model Integration, desarrollado por el Software Engineering Institute, Universidad de Carnegie Mellon, CMMI-DEV, v1.2, 2006. Este modelo se centra en la ingeniería de sistemas y el desarrollo de software, extiende las mejores prácticas de SW-CMM, SECM "Systems Engineering Capability Model" y IPD-CMM "Integrated Product Development Capability Maturity Model", y cumple completamente con los estándares más relevantes de ISO. En la categoría Ingeniería, incluye como áreas de proceso a la Gestión de Requisitos y al Desarrollo de Requisitos; [CMMI 06] [Ahern 03] [Chrissis 03].
- **SPICE**, Software Process Improvement and Capability dEtermination, desarrollado conjuntamente por varios centros técnicos en diferentes lugares del mundo, 1995. En su parte 2, define un modelo de gestión de procesos, donde incluye en la Categoría Procesos de Ingeniería el proceso Desarrollo de Requisitos de Software; http://www.sqi.gu.edu.au/spice/ [Rout 95].
- **Trillium**, desarrollado por Bell Canada, Northern Telecom and Bell-Northern Research, 1994. Este Modelo cubre todos los aspectos del proceso de desarrollo de software e incluye actividades relacionadas con marketing. Está basado en la versión 1.1 de SW-CMM, e incluye entre otros estándares: ISO 9001:1994, ISO 9000-3:1991, IEEE Software Engineering Standards Collection 1993 y IEC Standard Publication 300: 1984. La arquitectura de este modelo se basa en hojas de ruta[179] en vez de áreas claves como en CMM, e incluye a "Requirements Management" como una hoja de ruta; http://ricis.cl.uh.edu/trillium/trillium.html.
- **TickIT**, desarrollado por British Standards Institute, 1998. Es un esquema para la certificación y construcción de sistemas de gestión de calidad del software, conforme al estándar ISO 9001:2000, y diseñado específicamente para el desarrollo de software; http://www.tickit.org/
- **SECM (EIA-731)**, System Engineering Capability Model, 1999, de Electronics Industries Association. Este modelo surge de la fusión de dos modelos de la ingeniería de sistemas: SE-CMM (Systems Engineering Capability Maturity Model, 1994) y SECAM (Systems Engineering Capability Assessment Model, 1994). En este modelo se ubica como primera área de foco a "Define Stakeholder and System Level Requirements" dentro del

[179] Una *hoja de ruta* ("roadmap") es un conjunto de prácticas relacionadas que se enfocan en una área o necesidad organizacional, o un elemento específico dentro del proceso de desarrollo del producto.

dominio ”System Engineering Technical”.

- **ISO/IEC 12207-1**, Information technology - Software Life Cycle Processes, 1995. En este estándar, el proceso Desarrollo incluye la actividad Análisis de Requisitos. El estándar requiere que se establezcan baselines de los requisitos del software, del diseño del software y del código del software.
- **ISO 9000-3**, Quality management and quality assurance standards, 1997, Parte 3: guías para la aplicación de ISO 9001: 1994 al desarrollo, provisión y mantenimiento de software. Esta norma establece que dentro del plan de desarrollo de software se incluya el análisis de requisitos como una actividad técnica y la comunicación de requisitos como una actividad de gestión. También establece una clasificación de requisitos.
- **ISO/IEC 15026**, Information technology - System and software integrity levels, estándar para niveles de integridad en sistemas de software, 1998.
- **ISO/IEC TR 15271**, Information technology - Guide for ISO/IEC 12207. (Software life cycle processes), 1998.
- **ISO/IEC 15504**, Information technology - Software process assessment, 1998.

A.4. Congresos dedicados exclusivamente a esta disciplina

- **RE**, IEEE International Requirements Engineering Conference, congreso que se realiza anualmente desde 1993. Comenzó siendo dos eventos diferentes bianuales: RE (IEEE Intl. Symposium on Requirements Engineering) desde 1993 y ICRE (IEEE Intl. Conference on Requirements Engineering) desde 1994, que se unieron a partir del 2002.
- **REFSQ**, International Workshop on Requirements Engineering: Foundation for Software Quality, congreso anual desde 1995.
- **AWRE**, Australian Workshop on Requirements Engineering, congreso australiano que se realiza anualmente desde 1996.
- **WER**, Workshop en Ingeniería de Requisitos, congreso iberoamericano que se realiza anualmente desde 1998.
- **IDEAS**, Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes de Software, congreso iberoamericano que se realiza anualmente desde 1998.
- **ReConf**, Requirements Engineering Tagung, congreso alemán que se realiza anualmente desde 2002.

A.5. Revistas y libros dedicados exclusivamente al tema

- “*Requirements Engineering Journal*”, Springer-Verlag London Ltd., Gran Bretaña, ISSN: 0947-3602, desde 1996.
- “*Requirenautics Quarterly*”, revista electrónica de BCS Requirements Engineering Specialist Group, http://research.ivv.nasa.gov/~steve/resg/
- “*Software Requirements: Analysis and Specification*”, Davis, A.M., Prentice-Hall Intl., Englewood Cliffs, NJ, 2ª edición, 1993.
- “*System Requirements Engineering*”, Loucopoulos, P., Karakostas, V., Mc Graw-Hill, 1995.
- “*Software Requirements & Specifications*”, Jackson, M., Addison-Wesley,

1995.
- "*Software Requirements Engineering*", Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, 1997.
- "*Requirements Engineering Process and Techniques*", Kotonya, G., Sommerville, I. John Wiley and Sons, 1998.
- "*Practical Software Requirements: A manual of Content and Style*", B.L. Kovitz, Greenwich, CT: Manning Publications Co., 1998.
- "*Mastering the Requirements Process*", Susanne & James Robertson, Addison-Wesley Professional, 1º edición, 1999.
- "*Writing Better Requirements*", Alexander, I.F., Stevens, R., Addison-Wesley Professional, 1º edición, 2002.
- "*Software Requirements*", Wiegers, K., Microsoft Press, 2º edición, 2003.
- "*Managing Software Requirements - A Use Case approach*", D. Leffingwell y D. Widrig, Addison-Wesley Professional, 2º edición, 2003.
- "*Perspectives on Software Requirements*", J.C.S.P. Leite y J.H. Doorn, Kluwer Academic Publishers, 2004.
- "*The Requirements Engineering Handbook*", R.R. Young, Artech House, Norwood, MA, 2004.
- "*Requirements Engineering for Sociotechnical Systems*", J.L. Maté y A. Silva, Information Science Publishing, Londres, 2005.
- "*Requirements Engineering*", E. Hull, K. Jackson y J. Dick, Springer Science, EEUU, 2º edición, 2005.

A.6. Herramientas comerciales que soportan actividades de la IR

- **DOORS** (Dynamic Object Oriented Requirements System) de Telelogic, http://www.telelogic.com/products/doors/doors/
- **Requisite-Pro** de Rational Software, http://www.rational.com/
- **RTM** (Requirements & Traceability Management) de Serena Software Inc., http://www.serena.com/Products/rtm
- **EasyRM** (Easy Requirements Management) de Cybernetic Intelligence GmbH, http://www.easy-rm.com/
- **IRqA** (Integral Requisite Analizer) de TCP Sistemas & Ingeniería S.L., España, http://www.irqaonline.com
- **AnalystPro** de Goda Software Inc., http://www.analysttool.com
- **CARE** (Computer Aided Requirements Engineering) de SOPHIST Group, http://www.sophist.de/
- **CaliberRM** de Borland, http://www.borland.com/us/products/caliber/index.html/
- **GMARC** (Generic Model Approach to Requirements Capture) de Computer System Architects Ltd., http://www.freenetpages.co.uk/hp/csa/
- **Objectiver** de Cediti, http://www.objectiver.com/
- **RDT** (Requirements Design & Traceability) de Igatech, http://www.igatech.com/
- **RMTrack** de RBC Product Development, http://www.rbccorp.com/
- **VeroTrace** de Verocel, http://www.verocel.com/verotrace.htm
- **XTie-RT** (Cross Tie Requirements Tracer) de Teledyne Brown Engineering, http://www.tbe.com/

A.7. Comunidades informáticas

A.7.1. Internacionales

- **RESG**: the Requirements Engineering Specialist Group of the British Computer Society, http://www.resg.org.uk/
- **RENOIR**: the Requirements Engineering Network Of International cooperating Research groups, establecido dentro del programa Fourth Framework Programme (FP4) de la Unión Europea. Es una red de grupos de investigación, cada uno con una excelencia establecida en el área de IR, conocido colectivamente como ESPRIT, http://www.cs.ucl.ac.uk/research/renoir/
- **CREWS**: Cooperative Requirements Engineering with Scenarios, proyecto de investigación de largo término de ESPRIT, http://www.soi.city.ac.uk/~cc559/CREWS.html.

A.7.2. Nacionales

- Proyecto de la Universidad de Belgrano, iniciado en 1995 y finalizado en el 2000. Primera iniciativa en el país sobre el estudio de la Ingeniería de Requisitos.
- Proyecto de la Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, acreditado e incorporado en el sistema de Incentivos para Docentes Investigadores de las Universidades Nacionales, grupo proveniente del Proyecto Universidad de Belgrano, iniciado en el 2001 hasta el 2005.
- Proyecto de la Universidad Nacional de La Matanza, Departamento de Ingeniería e Investigaciones Tecnológicas, acreditado e incorporado en el sistema de Incentivos para Docentes Investigadores de las Universidades Nacionales, grupo proveniente de la Universidad Tecnológica Nacional, iniciado en el 2006.
- Instituto de Investigación y Transferencia en Tecnología Informática Avanzada (INTIA) de la Universidad Nacional del Centro de la Provincia de Buenos Aires, desde 1996.
- Grupo de Investigación en Paradigmas de Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional Santa Fe, desde 2001.
- Tesis y otros trabajos realizados a partir de la Maestría en Ingeniería de Software de la Universidad Nacional de La Plata, tesis aprobadas desde el 2001.
- Tesis y otros trabajos realizados a partir de la Maestría en Informática de la Universidad Nacional de La Matanza, tesis en desarrollo desde 2003.

Apéndice B
Publicaciones relacionadas con la tesis

«There is nothing permanent except change»

Heraclitus, 500 AC

B.1. Publicaciones en Capítulo de Libro

- "*Defining System Context using Scenarios*", Leite, J.C.S.P., Doorn, J.H., Kaplan, G.N., Hadad, G.D.S., Ridao, M.N., en el libro "Perspectives on Software Requirements", editor Kluwer Academic Publishers, Estados Unidos, ISBN: 1-4020-7625-8, capítulo 8, pp.169-199, 2004.

B.2. Publicaciones Internacionales con Referato

- "*Integración de escenarios con el léxico extendido del lenguaje en la elicitación de requerimientos: aplicación a un caso real*", Hadad, G., Kaplan, G., Oliveros, A, Leite, J.C.S.P., Revista de Informática Teórica y Aplicada (RITA), Vol.6, No.1, pp.77-103, Instituto de Informática da Universidade Federal do Rio Grande do Sul, Brasil, 1999.

- "*A Scenario Construction Process*", Leite, J.C.S.P., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., Requirements Engineering Journal, Vol.5, No.1, pp.38-61, Springer-Verlag London Ltd., Gran Bretaña, 2000.

- "*Scenario Inspections*", Leite, J.C.S.P., Doorn, J.H., Hadad, G.D.S., Kaplan, G.N., Requirements Engineering Journal, Vol.10, No.1, pp.1-21, Springer-Verlag London Ltd., Gran Bretaña, Enero 2005.

B.3. Publicaciones en Anales de Congresos

- "*Enfoque Middle-Out en la Construcción e Integración de Escenarios*", Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., Leite, J.C.S.P. , Anales de WER'99 - Workshop en Requerimientos, XXVIII JAIIO, Buenos Aires, Argentina, pp.79-94, Septiembre 1999.

- "*Inspección del Léxico Extendido del Lenguaje*", Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., Leite, J.C.S.P., WER'00 - Workshop de Engenharia de Requisitos, Río de Janeiro, Brasil, pp.70-91, Julio 2000.

- "*Comprendiendo el Universo de Discurso Futuro*", Doorn, J.H., Hadad, G.D.S., Kaplan, G.N., WER'02 - Workshop en Ingeniería de Requisitos,

Valencia, España, pp.117-131, Noviembre 2002.

- "*Construcción de Escenarios Futuros*", Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., Workshop de Investigadores en Ciencias de la Computación, WICC 2003, presentación de póster e inclusión en los anales electrónicos, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Argentina, Mayo 2003.

B.4. Publicaciones Internas con referato

- *"Detección de Discrepancias, Errores y Omisiones en Escenarios"*, Doorn, J., Kaplan, G., Hadad, G., Leite, J.C.S.P., Documento de Trabajo, Serie Programa de Formación en Investigación de Docentes Auxiliares, N° 15, Universidad de Belgrano, Buenos Aires, 1999, 24 páginas.

- *"Construcción de Escenarios: Top-Down o Bottom-Up?"*, Doorn, J., Hadad, G., Kaplan, G., Leite, J.C.S.P., Documento de Trabajo, Serie Programa de Formación en Investigación de Docentes Auxiliares, N° 17, Universidad de Belgrano, Buenos Aires, 1999.

B.5. Publicaciones sin referato

- *"Using Scenario Inspections on Different Scenarios Representations"*, Leite, J.C.S.P., Doorn, J.H., Hadad, G.D.S., Kaplan, G.N., Serie Monografias em Ciência da Computação, Nº33/03, Pontificia Universidad Católica do Rio de Janeiro, Brasil, 2003.

- *"Ingeniería de Requisitos"*, Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., Notas de Clase, Código K2ET1, Editorial del Centro de Estudiantes de la Facultad Regional Buenos Aires, UTN, Buenos Aires, Abril 2004, 90 páginas.

- *"Inspecciones"*, Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., Notas de Clase, Código K2ET2, Editorial del Centro de Estudiantes de la Facultad Regional Buenos Aires, UTN, Buenos Aires, Julio 2004, 47 páginas.

- *"Ingeniería de Requerimientos"*, Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., Notas de Clase, Código 635-3, Editorial Liga Federal Universitaria, Cátedra Ingeniería de Requerimientos, Departamento de Ingeniería e Investigaciones Tecnológicas, UNLaM, Prov. Buenos Aires, Mayo 2005, 92 páginas.

- *"Inspecciones"*, Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., Notas de Clase, Código 635-2, Editorial Liga Federal Universitaria, Cátedra Ingeniería de Requerimientos, Departamento de Ingeniería e Investigaciones Tecnológicas, UNLaM, Prov. Buenos Aires, Mayo 2005, 51 páginas.

- *"Modelos de Proceso de Software"*, Hadad, G.D.S., Kaplan, G.N., Doorn, J.H., Notas de Clase, Código 633-1, Editorial Liga Federal Universitaria,

Cátedra Proceso de Software, Departamento de Ingeniería e Investigaciones Tecnológicas, UNLaM, Prov. Buenos Aires, Septiembre 2005, 25 páginas.

B.6. Artículos enviados para publicación

- *"Creating Software System Context Glossaries",* Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., aceptado para su publicación en Encyclopedia of Information Science and Technology, Idea Group Publishing, 2º edición, 2007.

- *"Handling Extemporaneous Information in Requirements Engineering",* Kaplan, G.N., Doorn, J.H., Hadad, G.D.S., aceptado para su publicación en Encyclopedia of Information Science and Technology, Idea Group Publishing, 2º edición, 2007.

- *"Requisitos Candidatos Espontáneos"*, Kaplan, G.N., Doorn, J.H., Hadad, G.D.S., aceptado para su publicación en Revista Eletrônica de Sistemas de Informação, Brazil, 2007.

- *"Panorama de la Ingeniería de Requisitos"*, Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., aceptado para su publicación en la revista científica de la UNLaM, 2006.

- *"Información Extemporánea en el Proceso de Obtención de Requisitos"*, Kaplan, G.N., Doorn, J.H., Hadad, G.D.S., enviado a Revista Colombiana de Computación, 2006.

B.7. Artículos en etapa final de redacción

- *"Creating Application Domain Glossaries for Software Development"*, Leite, J.C.S.P., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., 2004.

- *"Explicitar Requisitos del Software usando Escenarios"*, Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., 2007.

- *"Visiones del Negocio con Casos de Uso"*, Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., 2007.

- *"Trazabilidad y Versionado de Requisitos usando modelos en lenguaje natural"*, Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., 2007.

Apéndice C
Trabajos derivados de esta tesis

«The purpose of war is not battle but victory.»
or:
The purpose of analysis is not modelling but understanding.

Sun Tsu, "The Art of War", 500 BC. Extraído de la página web de Ian Alexander.

C.1. Tesis relacionadas

- García, O.F., Gentile, C.G., "*Escenarios del proceso de construcción de escenarios: Autoaplicación de la metodología*", tesis de grado de la Carrera de Ingeniería de Sistemas, Facultad de Ciencias Exactas, UNICEN, Director: Prof. Jorge H. Doorn, Marzo 2000.

- Tornabene, S., Petersen, L., "*HEAR: Una Herramienta para la Adquisición de Requisitos*", tesis de grado de la Carrera de Ingeniería de Sistemas, Facultad de Ciencias Exactas, UNICEN, Director: Prof. Jorge H. Doorn,1999.

- Leonardi, M.C., "*Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural*", tesis de Maestría en Ingeniería de Software, Facultad de Informática, Universidad Nacional de La Plata, Noviembre 2001. Director: Prof. Julio Cesar Leite, PhD. (Pontificia Universidad Católica do Río de Janeiro), http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Leonardi.pdf, accedida el 24-02-2006.

- Ridao, M., "*Uso de Patrones en el Proceso de Construcción de Escenarios*", tesis de Maestría en Ingeniería de Software, Facultad de Informática, Universidad Nacional de La Plata, Noviembre 2001. Director: Prof. Jorge H. Doorn (Universidad del Centro de la Provincia de Buenos Aires), http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Ridao.pdf, accedida el 24-02-2006.

- Antonelli, L., "*Traceability en la elicitación y especificación de requerimientos*", tesis de Magíster en Ingeniería de Software, Facultad de Informática, UNLP, Director: Prof. Alejandro Oliveros, Mayo 2003, http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Antonelli.pdf, accedida el 24-02-2006.

- Gil, G.D., "*Herramienta para Implementar LEL y Escenarios (TILS)*", tesis de Magíster en Ingeniería de Software, Facultad de Informática, UNLP, Director: Prof. Alejandro Oliveros, Enero 2002, http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Gil.pdf, accedida

el 24-02-2006.

- Bertolami, M., "*Una propuesta de Análisis de Puntos de Función aplicado a LEL y Escenarios*", tesis de Magíster en Ingeniería de Software, Facultad de Informática, UNLP, Director: Prof. Alejandro Oliveros, Octubre 2003, http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Bertolami.pdf, accedida el 24-02-2006.

- Centeno, M.E., "*Estimación del tamaño de los artefactos producidos en la elicitación de requerimientos*", tesis de Magíster en Ingeniería de Software, Facultad de Informática, UNLP, Director: Prof. Alejandro Oliveros, Abril 2004, http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Centeno.pdf, accedida el 24-02-2006.

- Zuñiga, J.A., "*Proceso LEC: un proceso definido para aplicar LEL, escenarios y CRC*", tesis de Magíster en Ingeniería de Software, Facultad de Informática, UNLP, Director: Prof. Alejandro Oliveros, Abril 2004, http://journal.info.unlp.edu.ar/postgrado/Carreras/Magister/Tesis/Zuñiga.pdf, accedida el 24-02-2006.

- Mauco, M.V., "*A Technique for an Initial Specification in RSL*", tesis de Magíster en Ingeniería de Software, Facultad de Informática, UNLP, Director: Prof. Daniel Riesco, Julio 2004.

Apéndice D
Aplicaciones de la Estrategia

«Requirements … are a major source of expensive bugs. The range is from a few percent to more than 50% … What hurts most ... is that they're the earliest to invade the system and the last to leave. It's not unusual for a faulty requirement … only to be caught after hundreds of sites have been installed. »

B. Beizer , "Software Testing Techniques", 2° ed., Van Nostrand Reinhold, NY, 1990

D.1. Introducción

La justificación del proceso expuesto en esta tesis fue presentada inicialmente en Leite et al. [Leite 97] y los detalles del primer caso de estudio, basado en el *"Sistema Nacional para la Obtención de Pasaportes"* se informaron en Hadad et al. [Hadad 99]. La estrategia consistía en construir el LEL del UdeD. Luego, se construían los escenarios basados en el UdeD y usando el vocabulario definido el LEL. El léxico y los escenarios se validaban posteriormente con los usuarios y los escenarios se usaban para verificar el léxico.

En esta tesis se presenta como ejemplo una variación del ampliamente conocido *"Sistema de Agenda de Reuniones"*. El caso se estudió solamente para el proceso de IR y consistió en modelar la agenda de reuniones para las autoridades de la Universidad de Belgrano - Buenos Aires, aplicando SDRES.

Se construyó un LEL de 34 entradas, desarrollado por dos investigadores. Mientras que se generaron de este sistema tres conjuntos diferentes de escenarios actuales con 16, 13 y 17 escenarios, dos conjuntos fueron desarrollados por estudiantes de grado y uno por un equipo de investigadores. A partir de este último conjunto se desarrollaron los escenarios futuros y la lista de requisitos. La Universidad de Belgrano se utilizó como organización cliente basándose en el caso de estudio propuesto por [van Lamsweerde 93].

Cabe destacar que se ha construido un LEL y un conjunto de escenarios para los procesos de construcción del LEL y de los escenarios actuales [García 99]. EL LEL constó de 78 símbolos y el conjunto de 64 escenarios, producido por un grupo de estudiantes de grado supervisado por investigadores. Este léxico describe todos los puntos importantes en la construcción del LEL y de los escenarios. El conjunto de escenarios describe las situaciones que ocurren durante la construcción del léxico y de los escenarios. Tanto el LEL como el conjunto de escenarios fueron analizados por el autor de la tesis junto con otros investigadores del grupo. El léxico y los escenarios están disponibles en http://usuarios.arnet.com.ar/ogarcia. No refleja la versión corregida y ampliada presentada en esta tesis pero sirve de base para todos aquellos estudiantes y

practicantes que deseen interiorizarse sobre esta estrategia.

Las siguientes sub-secciones presentan los productos de aplicar SDRES para el caso del *"Sistema de Agenda de Reuniones"* realizado por el grupo de investigadores.

D.2. Ejemplo de Léxico Extendido del Lenguaje

Las entrevistas fueron realizadas por dos entrevistadores que cumplieron diferentes roles. El entrevistador A tomó nota textual de lo que el cliente decía. El entrevistador B anotó sólo las palabras o frases que el cliente repetía o las que consideró importantes teniendo en cuenta el contexto y el énfasis que el cliente ponía en ellas. El entrevistador A obtuvo una lista de símbolos candidatos del texto producido en cada entrevista. Luego las unió y formó la única lista candidata de símbolos A. El entrevistador B juntó las listas de símbolos que obtuvo durante las entrevistas y formó la lista candidata de símbolos B. Se verificaron las listas candidatas A y B, y se comprobó que en un 75% aproximadamente coincidían. El 25% restante correspondían a símbolos obtenidos por el entrevistador B que no detectó A y viceversa. Se formó una lista C con los símbolos que coincidían en ambas listas, más tarde se chequeó con los clientes el 25% que había quedado afuera. Algunos se descartaron, otros se incluyeron como símbolos nuevos y otros se detectaron como sinónimos. La lista candidata contó con 35 símbolos, quedando finalmente un LEL de 34 símbolos.

LISTA DE SÍMBOLOS DEL LEL

1. agenda / agenda de reuniones (O)
2. ámbito para la reunión / lugar físico de la reunión / espacio físico / lugar (O)
3. anulación de reunión / cancelación de reunión / aviso de cancelación (V)
4. aviso de concurrencia / confirmación de asistencia (V)
5. aviso de no concurrencia (V)
6. cambios en los requerimientos de la reunión (V)
7. convocado / gente citada / gente convocada (S)
8. convocante (S)
9. convocatoria / cita (V)
10. cronograma de reuniones / calendario (O)
11. desconvocatoria (V)
12. diseño de la agenda de reuniones (V)
13. disponibilidad de espacio (V)
14. esquema / esquema de base (O)
15. fecha de la reunión / fecha de convocatoria / fecha establecida (O)
16. fijación de horarios / establecer fechas (V)
17. hora de la reunión / horario (O)
18. horarios disponibles (O)
19. listado de notificación / registro de notificación / listado de convocados / listado para convocatoria (O)
20. material a presentar / material para reuniones / material de exposición (O)

21. material físico (O)
22. material para repartir (O)
23. objetivo (O)
24. organización de reuniones / organizar la reunión (V)
25. participante / integrante (S)
26. recordatorio (V)
27. reemplazante (S)
28. requerimientos (O)
29. reunión / evento (V)
30. secretaria (S)
31. tema / contenidos / puntos centrales (O)
32. temario (O)
33. tiempo de traslado a la reunión (O)
34. traslado de fecha (V)

Siendo: (O) – Objeto (S) – Sujeto (V) – Verbo (E) – Estado

DESCRIPCIÓN DE SÍMBOLOS

Agenda / Agenda de Reuniones
Noción:
– Elemento donde se registran los *requerimientos* de las *reuniones*.
Impacto:
– Se registra el *objetivo* de la *reunión*, los *convocados*, los *reemplazantes*, la *fecha de la reunión*, la *hora de la reunión*, el *lugar físico de la reunión*, *material para repartir*, *material a presentar*, *material físico* y *tiempo de traslado a la reunión*, al *organizar la reunión*. – Se registra el *objetivo* de la *reunión*, la *fecha de la reunión*, la *hora de la reunión*, el *lugar físico de la reunión*, *material a presentar*, *material para repartir* y *tiempo de traslado a la reunión*, al recibir la *convocatoria* a una *reunión*. – Se registran los *cambios en los requerimientos de la reunión*. – Se registran los *traslados de fechas*. – Se registran las *anulaciones de reuniones*. – Se la consulta para conocer los *horarios disponibles*. – Se la consulta para conocer las *reuniones*. – Se registra la información del *cronograma de reuniones*.

Ámbito Para La Reunión / Lugar Físico De La Reunión / Espacio Físico / Lugar
Noción:
– Sitio donde se lleva a cabo la *reunión*, según la *disponibilidad de espacio*. – Se establece junto con la *fecha de la reunión* y la *hora de la reunión*.
Impacto:
– Se registra en la *agenda*. – Se registra en el *temario*. – Se informa al realizar la *convocatoria*.

Anulación De Reunión/ Cancelación De Reunión / Aviso De Cancelación
Noción:
– Comunicación que establece el *convocante* o la *secretaria* a todos los *convocados* informando que la *reunión* fue cancelada. – Debe realizarse antes de *fecha de la reunión*. – Se puede utilizar el *listado de convocados* para realizar la cancelación de la reunión. – Puede ser por escrito, verbal, fax, telefónico, etc.
Impacto:
– Se registra en la *agenda*. – Se registra en el *cronograma de reuniones*. – Se anula la reserva de *espacio físico*. – Se anula la reserva de *material físico*.

Aviso De Concurrencia / Confirmación De Asistencia
Noción:
– Comunicación que establece un *convocado* con el *convocante* de la reunión o *secretaria* para confirmar su participación o la asistencia de un *reemplazante*. – Debe realizarse antes de la *fecha de la reunión*. – Se avisa por teléfono, por fax, por escrito o personalmente al sitio establecido para la confirmación.
Impacto:
– Se registra en la *agenda*. – Se registra en el *listado de convocados*.

Aviso De No Concurrencia
Noción:
– Comunicación que establece un *convocado* con el *convocante* o la *secretaria* para informar que no asistirá a la *reunión*. – Debe realizarse antes de *fecha de la reunión*. – Puede ser por escrito, verbal, fax, teléfono, etc.
Impacto:
– Se registra en la *agenda*. – Se registra en el *listado de convocados*.

Cambios En Los Requerimientos De La Reunión
Noción:
– Comunicación que establece el *convocante* o su *secretaria* para informar a los *convocado***s** el cambio del *lugar físico de la reunión*, *temario*, *material físico*, *material a presentar*, *material para repartir*. – Debe realizarse antes de la *fecha de la reunión***.** – Se avisa por teléfono, fax, escrito o personalmente. – Se puede utilizar el *listado de convocados*.
Impacto:
– Se avisa a todos los *convocados*. – Se registra en la *agenda***.** – Se modifica la reserva de *espacio físico*. – Se modifica la reserva de *material físico*.

Convocado / Gente Citada / Gente Convocada
Noción:
– Persona invitada a la *reunión*.
Impacto:
– Puede informar sus *horarios disponibles*. – Puede dar el *aviso de concurrencia*. – Debe estar en el *lugar*, *fecha establecida* y *horario* establecidos en la *convocatoria*. – Puede dar el *aviso de no concurrencia*. – Asigna un *reemplazante* en caso de no poder asistir y lo informa al *convocante* o *secretaria*. – Puede definir el *material para repartir*. – Registra el *tiempo de traslado a la reunión*.

Convocante
Noción:
– Persona que invita a los *convocados* a una *reunión*. – Puede ser un *participante*.
Impacto:
– Determina el *objetivo* de la *reunión*, los *temas* a tratar, los *convocados*, el *material para repartir* y el *material a presentar*. – Registra el *objetivo* y los *convocados*, en el *esquema de base*. – Realiza el *diseño de la agenda de reuniones*. – *Organiza la reunión*. – Registra en la *agenda* al *reemplazante*. – Determina la *anulación de la reunión*. – Determina el *cambio en los requerimientos de la reunión*. – Determina el *traslado de fecha*. – Determina la *desconvocatoria* de ser necesario.

Convocatoria / Cita
Noción:
– Comunicación que establece el *convocante* o la *secretaria* para informar a todos los *convocados* la realización de la *reunión*. – La *convocatoria* se realiza una vez definida la *fecha de la reunión*, *horario* y *lugar físico de la reunión*. – Puede ser por escrito, verbal, fax, teléfono, etc. – Para realizar la *convocatoria*, se utiliza la *agenda* o el *listado para convocatoria*.
Impacto:
– Se puede entregar un *temario*. – Se informa a cada *convocado* sobre los *requerimientos* de la *reunión*.

Cronograma De Reuniones / Calendario
Noción:
– Diagrama donde se establece, para un lapso, todas las *reuniones* programadas. – Se especifican las *fechas de las reuniones*.
Impacto:
– Se transcribe a la *agenda*. – Se incluyen las nuevas *reuniones*. – Se registran los *traslados de fechas*. – Se registran las *anulaciones de reuniones*.

Desconvocatoria
Noción:
– Comunicación que establece el *convocante* de la *reunión* o su *secretaria* para avisar a los *participantes* que hubo un *traslado de fecha*, *anulación de la reunión* o *cambio en los requerimientos de la reunión*. – Debe realizarse antes de la *fecha de la reunión*. – Se puede utilizar el *listado para convocatoria* para realizar la *desconvocatoria*. – Se avisa por teléfono, por fax, por escrito o personalmente a los *participantes*.
Impacto:
– Se registra en la *agenda*. – Se registra en el *cronograma de reuniones*, en el caso de *traslado de fecha* o *anulación de la reunión*. – Se puede anular la reserva de *espacio físico*. – Se puede anular la reserva de *material físico*.

Diseño De La Agenda De Reuniones
Noción:
– Actividad que realiza el *convocante* para determinar la *fecha de la reunión*, el *horario* y el *lugar* de las *reuniones*, basándose en los *horarios disponibles* de los *convocados*, la *disponibilidad de espacio* y otras *reuniones* ya registradas en la *agenda*. – Tiene la finalidad de organizar los tiempos y evitar superposiciones y omisiones de *reuniones*.
Impacto:
– Se registra en la *agenda*: el *objetivo*, la *fecha establecida*, el *horario* y el *lugar* establecido para llevar a cabo la *reunión*. – Se confecciona el *listado para convocatoria*. – Se confecciona el *temario*. – Se registra en el *cronograma de reuniones*. – Se reserva el *espacio físico*. – Se reserva el *material físico*.

Disponibilidad De Espacio
Noción:
– Verificación de espacio libre en la *fecha establecida* y la *hora de la reunión*.
Impacto:
– Se reserva el *lugar* elegido. – Se determina el *lugar físico de la reunión*.

Esquema / Esquema De Base
Noción:
– Diagrama de las *reuniones* para todo un período. – No tiene *fechas establecidas*. – Cuando se *establecen las fechas*, el esquema se convierte en el *cronograma de reuniones*.
Impacto:
– Se registran las posibles *reuniones*.

Fecha De La Reunión / Fecha De Convocatoria / Fecha Establecida
Noción:
– Fecha en la que se realiza la *reunión***.** – Debe ser establecida con anticipación a la *reunión*. – Se establece en la *fijación de horarios*. – Está comprendida dentro de los *horarios disponibles* de los *convocados*.
Impacto:
– Se registra en la *agenda*. – Se informa a los *convocados*. – Se registra en el *temario*.

Fijación De Horarios / Establecer Fechas
Noción:
– El *convocante* determina la *fecha de reunión* y la *hora de reunión*. – Se determina en función de los *horarios disponibles* de los *convocados*. – Se determina en función de la *disponibilidad de espacio*.
Impacto:
– Se informa a los *convocados* la *fecha establecida* y la *hora de la reunión*. – Se registra en la *agenda* la *fecha de la reunión* y la *hora de la reunión*.

Hora de la Reunión / Horario
Noción:
– Hora en que se inicia la *reunión*. – Debe ser establecida con anticipación a la *reunión*. – Se establece en la *fijación de horarios*. – Comprendido dentro de los *horarios disponibles* de los *convocados*, de ser posible.
Impacto:
– Se registra en la *agenda*. – Se registra en el *temario*. – Se informa al realizar la *convocatoria*.

Horarios Disponibles
Noción:
– Fechas y horas libres en la *agenda*.
Impacto:
– Se utiliza en la *fijación de horarios*.

Listado De Convocados / Listado Para Convocatoria / Listado De Notificación / Registro De Notificación
Noción:
– Lista con los datos de los *convocados* a una reunión.
Impacto:
– Se registra la *convocatoria*. – Se registra la *confirmación de asistencia* del *convocado*. – Se registra el *aviso de no concurrencia* del *convocado*. – Se registra el *recordatorio* a los *convocados*. – El *convocado* notifica que fue citado firmando en la lista. – Se utiliza para realizar la *desconvocatoria*. – Se registra el *reemplazante*.

Material A Presentar / Material Para Reuniones / Material de Exposición
Noción:
– Material que el *participante* prepara y lleva a la *reunión* para exponer. – Puede ser: transparencias, gráficos, etc.
Impacto:
– Se registra en la *agenda*. – Se expone en la *reunión*.

Material Físico
Noción:
– Accesorios que se utilizan en la *reunión*. – Puede ser un proyector, una computadora, una pizarra, etc.
Impacto:
– Se registra en la *agenda*. – Se reserva con anticipación a la *reunión*. – Se lleva a la *reunión*.

Material Para Repartir
Noción:
– Material que el *participante* prepara y lleva a la *reunión* para entregar a los otros *participantes*. – Puede ser: informes, planillas, etc.
Impacto:
– Se registra en la *agenda*. – Se entrega a todos los *participantes*.

Objetivo
Noción:
– Finalidad de la *reunión*. – Resultado esperado de la *reunión*.
Impacto:
– Se registra en la *agenda*. – Puede registrarse en el *temario*.

Organización De Reuniones / Organizar La Reunión
Noción:
– Actividad de implementación de la *reunión*. – Su finalidad es el manejo de elementos para asegurar la realización de la *reunión*. – Tarea realizada por la *secretaria*, después del *diseño de la agenda de reuniones*.
Impacto:
– Se realiza la *convocatoria* a la *reunión*. – Se asegura el *material físico*. – Se asegura el *espacio físico*. – Se realiza el *recordatorio* de la *reunión* a los *convocados*.

Participante / Integrante
Noción:
– Persona que asiste a una *reunión*. – Puede ser *convocado* o *convocante*. – Puede exponer en la *reunión*.
Impacto:
– *Confirma la asistencia* a la *reunión*. – Lleva el *material para repartir*. – Lleva el *material a presentar*. – Puede solicitar al *convocante* o la *secretaria* el *material físico* para la *reunión*. – Debe estar en el *lugar*, *fecha de la reunión* y *horario* establecidos en la *convocatoria*. – Registra en la *agenda* el *tiempo de traslado a la reunión*.

Recordatorio
Noción:
– Comunicación que establece el *convocante* o su *secretaria* para recordar a los *convocados* la *fecha de la reunión*, *horario* y *lugar* de la *reunión*. – Se informa unos días antes de la *fecha de la reunión*. – Se realiza por teléfono, fax, escrito o personalmente. – Se puede utilizar el *listado de convocados*.
Impacto:
– Se avisa a cada *convocado* sobre la *reunión*.

Reemplazante
Noción:
– Persona que asiste a una *reunión* en reemplazo de un *convocado*. – Es designado por el *convocado*.
Impacto:
– Debe estar en el *lugar*, *fecha de la reunión* y *horario* establecidos en la *convocatoria*. – Puede dar el *aviso de no concurrencia*.

Requerimientos
Noción:
– Elementos necesarios para realizar la *reunión*. – Pueden ser: *temario*, *espacio físico*, *material físico*, *material para repartir*, *material a presentar*, *fecha establecida*, *horario*, *gente citada* y confirmada.
Impacto:
– Si no se cumple con todos o algunos de ellos puede ocurrir: el *traslado de fecha* o la *anulación de la reunión*. – Se debe informar a los *convocados* cuando se produce un *cambio en los requerimientos de la reunión*. – Se registran en la *agenda*.

Reunión / Evento
Noción:
– Asamblea de gente con un *objetivo*. – Tiene un *lugar*, *fecha establecida* y *horario* establecido en la *agenda*. – Puede figurar en un *cronograma de reuniones*. – Puede tener un *temario*.
Impacto:
– Los *participantes* exponen los *temas* asignados con su *material a presentar*. – Los *participantes* entregan el *material para repartir*.

Secretaria
Noción:
– Persona que colabora con el *convocante* en la preparación y realización de las *reuniones*.
Impacto:
– *Organiza la reunión*. – Registra la *reunión* en la *agenda*. – Confecciona el *listado para convocatoria*. – Realiza la *convocatoria* a la *reunión* por indicación del *convocante*. – Registra en la *agenda* el *reemplazante*. – Puede reservar el *lugar físico de la reunión*. – Realiza la *desconvocatoria* a los *convocados*. – Realiza el *recordatorio* de la *reunión* a los *convocados*.

Tema / Contenidos / Puntos Centrales
Noción:
– Puntos más importantes a tratar en la *reunión*, definidos por el *convocante*.
Impacto:
– Se registra en el *temario*.

Temario
Noción:
– Papel que puede contener el *objetivo* de la *reunión*, los *temas* a tratar, los *participantes*, la *fecha establecida* y *horario*, el *lugar* donde se realizará. – Se confecciona una vez concluida la *organización de la reunión*. – Es una guía de contenidos de la *reunión*.
Impacto:
– Se entrega a los *participantes* antes de la *fecha de la reunión*.

Tiempo De Traslado A La Reunión
Noción:
– Tiempo que utiliza el *participante* para trasladarse hasta el *lugar* de la *reunión*.
Impacto:
– Se calcula de acuerdo al sitio de donde parte el *convocado* para asistir a la *reunión* en función de la *hora de la reunión*. – Se registra en la *agenda* la hora de partida hacia la *reunión*, teniendo en cuenta el tiempo de traslado.

Traslado De Fecha
Noción:
– Comunicación que establece el *convocante* o su *secretaria* para informar a los *convocados* el cambio de la *fecha de la reunión* a una nueva fecha. – Debe realizarse antes de la *fecha de la reunión***.** – Se avisa por teléfono, fax, escrito o personalmente. – Se puede utilizar el *listado de convocados*.
Impacto:
– Se avisa a todos los *convocados*. – Se registra en la *agenda***.** – Se registra en el *cronograma de reuniones***.** – Se anula la reserva de *espacio físico*. – Se anula la reserva de *material físico*.

D.3. Ejemplo de Escenarios Actuales

La Tabla D-1 resume la evolución del conjunto de escenarios durante el proceso de construcción para el *"Sistema de Agenda de Reuniones"*, donde cabe destacar el resultado positivo del proceso de derivación frente al resultado final del conjunto de escenarios. La Figura D-1 muestra gráficamente estos números.

Evolución de escenarios:
- ✓ 23 escenarios en la lista inicial
- ✓ 17 escenarios en la lista final (sin considerar el escenario integrador)
- ✓ 14 escenarios se mantuvieron de la lista inicial a la lista final
- ✓ 9 escenarios de la lista inicial se incluyeron como episodios simples dentro de otros escenarios
- ✓ 2 escenarios de la lista final fueron generados después de los procesos de V&V (estos dos escenarios se incluyeron previamente como episodios simples dentro de otros escenarios)
- ✓ 1 operación de Factorizar: generó 1 sub-escenario
- ✓ 1 operación de Consolidar: compuso 1 escenario excepción a partir de 2 escenarios excepción.
- ✓ 1 escenario integrador
- ✓ 87% de los escenarios se obtuvieron aplicando las heurísticas de derivación.

La composición de los 18 escenarios finales:
- ⇨ 4 escenarios
- ⇨ 9 sub-escenarios
- ⇨ 4 escenarios excepción
- ⇨ 1 escenario integrador

Tabla D-1. Evolución de escenarios actuales para el Sistema de Agenda de Reuniones

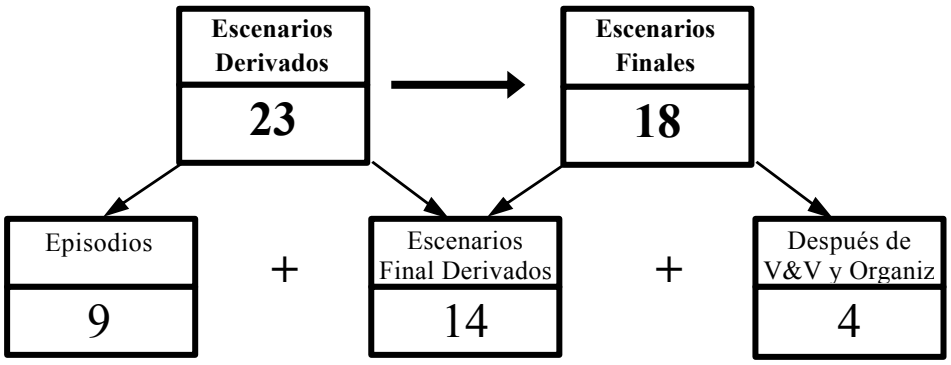


Figura D-1. Resultados del proceso de derivación y organización para el Sistema de Agenda de Reuniones

LISTA DE ESCENARIOS ACTUALES

Escenarios derivados de las entradas del LEL: CONVOCANTE y SECRETARIA

- ⇨ Requerir una reunión
- ⇨ Diseñar la agenda de reuniones
- ⇨ Organizar la reunión
- ⇨ Convocar a la reunión
- ⇨ Recordar la reunión
- ⇨ Anular la reunión
- ⇨ Trasladar la fecha de la reunión
- ⇨ Cambiar los requerimientos de la reunión

Escenarios derivados de las entradas del LEL: CONVOCADO, PARTICIPANTE y REEMPLAZANTE

- ⇨ Solicitar horarios disponibles
- ⇨ Avisar la concurrencia
- ⇨ Asistir a la reunión
- ⇨ Avisar la no concurrencia
- ⇨ Solicitar material físico

Nuevos sub-escenarios para ampliar información de episodios simples:

- ⇨ Establecer la fecha de la reunión
- ⇨ Establecer la nueva fecha de la reunión
- ⇨ Generar el listado para convocatoria
- ⇨ Reservar espacio físico en edificio anexo

DESCRIPCION DE ESCENARIOS ACTUALES

Escenarios Integradores
TITULO: **SISTEMA DE *AGENDA DE REUNIONES***
OBJETIVO: Mantener actualizadas los datos de las *reuniones*.
CONTEXTO: Ubicación Geográfica: -- Ubicación Temporal: -- Precondición: Debe existir un asunto a tratar o resolver por más de una persona..
RECURSOS:
ACTORES:
EPISODIO:

1. REQUERIR UNA *REUNIÓN*.
2. DISEÑAR LA *AGENDA DE REUNIONES*.
3. ORGANIZAR LA *REUNIÓN*.
4. ASISTIR A LA *REUNION*.

EXCEPCIONES:

Escenarios

TÍTULO: **REQUERIR UNA *REUNIÓN***

OBJETIVO:
Determinar el *objetivo* de la *reunión*, los *temas* y los *convocados*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe existir un asunto a tratar o resolver por más de una persona.

RECURSOS:
Agenda

ACTORES:
Convocante
Secretaria

EPISODIOS:
1. El *convocante* define la necesidad de una *reunión*, en base a una demanda externa o propia.
2. El *convocante* determina el *objetivo* de la *reunión*, los *temas* a tratar y los *convocados*.
3. **Si** se requieren varias *reuniones***,** el *convocante* arma un *esquema de base*.
4. **#** **[**El *convocante* determina el *material a presentar* por los *convocados*.**]**
5. **[**El *convocante* determina el *material para repartir*.**]** **#**
6. El *convocante* o la *secretaria* registra en la *agenda* el *objetivo*, los *temas*, los *convocados*, el *material a presentar* y el *material para repartir*.

EXCEPCIONES:

TÍTULO: **DISEÑAR LA *AGENDA DE REUNIONES***

OBJETIVO:
Determinar la *fecha de convocatoria*, la *horario*, el *lugar* y *requerimientos* de las *reuniones*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe presentarse previamente la necesidad de una *reunión*.

RECURSOS:
Agenda
cronograma de reuniones
esquema de base
horarios disponibles

ACTORES:
Convocante
Secretaria
Convocados

EPISODIOS:
1. **[**El *convocante* obtiene los datos de la *reunión* a diseñar del *esquema de base*.**]**
2. **# Si** no se tienen registrados los *horarios disponibles* de los *convocados*, SOLICITAR *HORARIOS DISPONIBLES*.
3. El *convocante* consulta sobre la *disponibilidad de espacio*.
4. **[**El *convocante* consulta sobre la disponibilidad de *material físico*.**]**
5. El *convocante* consulta la *agenda* para ver sus *horarios disponibles* en la misma.#
6. ESTABLECER LA *FECHA DE LA REUNION*.
7. # El *convocante* o la *secretaria* registra en la *agenda*: la *fecha establecida*, *horario* y *lugar*.
8. **[**El *convocante* define el *temario*.**]**
9. GENERAR EL *LISTADO PARA CONVOCATORIA*.
10. El *convocante* o la *secretaria* registra la *reunión* en el *cronograma de reuniones*.
11. El *convocante* o la *secretaria* reserva el *espacio físico*.
12. **[**El *convocante* o la *secretaria* reserva el *material físico*.**]**
13. **[**El *convocante* o la *secretaria* registra el *material físico* en la *agenda*.**]** #

EXCEPCIONES:

TÍTULO: ***ORGANIZAR LA REUNIÓN***

OBJETIVO:
asegurar el desarrollo eficiente de la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe realizarse previamente el *diseño de la agenda de reuniones*.

RECURSOS:
material físico

espacio físico

ACTORES:

Convocante
Secretaria
Convocados

EPISODIOS:

1. El *convocante* instruye a la *secretaria* sobre la *convocatoria* a la *reunión*.
2. CONVOCAR A LA *REUNIÓN*.
3. # AVISAR LA CONCURRENCIA.
4. AVISAR LA NO CONCURRENCIA.
5. **[**SOLICITAR *MATERIAL FÍSICO*.**]**
6. **[**RECORDAR LA *REUNIÓN*.**]**
7. **[**La *secretaria* asegura el *material físico* para la *fecha de la reunión*.**]**
8. La *secretaria* asegura el *espacio físico* para la *fecha de la reunión*. #

EXCEPCIONES:

- Imposibilidad de realizar la *reunión*. (ANULAR LA *REUNIÓN*.)
- Problemas con la *fecha de la reunión*. (TRASLADAR LA *FECHA DE LA REUNIÓN*.)
- Se requiere el cambio de algún *requerimiento* para la *reunión*. (CAMBIAR LOS *REQUERIMIENTOS* DE LA *REUNIÓN*.)

TÍTULO: **ASISTIR A LA *REUNIÓN***

OBJETIVO:

Asistir a la *reunión* según lo establecido en la *convocatoria*.

CONTEXTO:

Ubicación Geográfica: Se realiza en el *lugar físico de la reunión*.
Ubicación Temporal: se realiza en la *fecha establecida* y a la *hora de la reunión*.
Precondición:

El *convocado* o *reemplazante* debe haber confirmado su concurrencia, si fue solicitado.
El *convocado* o *reemplazante* llega al *lugar físico de la reunión* para participar en la misma.

RECURSOS:

listado de convocados
material para repartir
material a presentar
espacio físico

ACTORES:

secretaria
convocante
participante

EPISODIOS:

1. El *participante* llega al *lugar de la reunión* en la *fecha establecida* y *horario* establecido en la *convocatoria*.
2. La *secretaria* o el *convocante* registra la asistencia del *participante* en el *listado de convocados*. RESTRICCIÓN: Debe figurar en el *listado de convocados*.
3. # **[**El *participante* lleva el *material para repartir* a todos los *participantes*.**]**
4. **[**El *participante* lleva el *material a presentar*. RESTRICCIÓN: debe ser requerido antes de la *fecha de la reunión*.**]#**

EXCEPCIONES:

- El *participante* no puede llegar a la *reunión*. Episodio: 1. (La *secretaria* o *convocante* registra la ausencia del *participante* en el *listado de convocados* al finalizar la *reunión*)
- Asiste una persona no invitada en reemplazo de un *convocado*. Episodio: 2. (El *convocante* determina la participación o no de dicha persona y lo registra en el *listado de convocados*)

Sub-Escenarios

TÍTULO: **SOLICITAR *HORARIOS DISPONIBLES***

OBJETIVO:

Conocer los *horarios disponibles* de los *convocados*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:

Debe presentarse previamente la necesidad de una *reunión*, donde se determinaron los *convocados* a la misma.
No se tienen registrados los *horarios disponibles* de los *convocados* o se desean actualizar.

RECURSOS:

agenda

ACTORES:

convocado
secretaria
convocante

EPISODIOS:

1. La *secretaria* o el *convocante* solicita al *convocado* que le informe sus *horarios disponibles*.
2. El *convocado* informa su disponibilidad en base a las fechas y horas libres de su *agenda*.

3. La *secretaria* o el *convocante* toma nota.

EXCEPCIONES:

TITULO: **ESTABLECER LA *FECHA DE LA REUNIÓN***

OBJETIVO:
Fijar la *fecha de la reunión*, *horario* y *lugar físico de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe obtenerse previamente información de *horarios disponibles* de los *convocados* y del *convocante***,** y la *disponibilidad de espacio* y disponibilidad del *material físico*.

RECURSOS:
horarios disponibles
espacios disponibles
material físico disponible

ACTORES:
convocante

EPISODIO:
1. El *convocante* verifica la existencia de fechas comunes disponibles entre los *horarios disponibles* de los *convocados* y los suyos.
2. **Si** existen fechas comunes disponibles**,** el *convocante* selecciona la mejor fecha y hora según un criterio dado (la más cercana posible, la más alejada, en horario de oficina, etc.).
3. El *convocante* selecciona el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* para la *fecha de la reunión* establecida y la disponibilidad de *material físico* requerido.

EXCEPCIONES:
- Conflictos en los *horarios disponibles* de los *convocados*. Episodio 2. (ANULAR LA *REUNIÓN*)
- Conflictos en la *disponibilidad de espacio*. Episodio: 3. (RESERVAR *ESPACIO FISICO* EN EDIFICIO ANEXO)
- Conflicto en la disponibilidad de *material físico*. Episodio: 3. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial)

TITULO: **GENERAR EL *LISTADO PARA CONVOCATORIA***

OBJETIVO:
Obtener el *listado para convocatoria*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe establecerse previamente la *fecha de la reunión*.

RECURSOS:
archivo o agenda telefónica

ACTORES:
convocante
secretaria

EPISODIO:
1. La *secretaria* o el *convocante* obtiene información personal (domicilio, teléfono, e-mail) de cada *convocado*, proveniente de archivos o agenda telefónica.
2. La *secretaria* o el *convocante* confecciona una planilla con los datos de las personas a convocar.

EXCEPCIONES:

TÍTULO: **CONVOCAR A LA *REUNIÓN***

OBJETIVO:
Invitar a los *convocados* a una *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe realizarse previamente el *diseño de la agenda de reuniones*.

RECURSOS:
listado para convocatoria
copias del *temario*
medios de comunicación (teléfono, fax, correo, computadora, contestador automático, e-mail, etc.)

ACTORES:
secretaria
convocados

EPISODIOS:
1. La *secretaria* avisa, a través de algún medio de comunicación, a cada *convocado* la *fecha establecida*, *horario* y *lugar* de la *reunión*, utilizando el *listado para convocatoria*. RESTRICCIÓN: debe realizarse antes de la *fecha de la reunión*.
2. **[**La *secretaria* pide la *confirmación de asistencia* a la *reunión*.**]**
3. **Si** se trata de una *convocatoria* personal**,** solicita al *convocado* que firme la

notificación en el *listado para convocatoria*.

4. **[**La *secretaria* entrega/envía una copia del *temario* a cada *convocado***]**.
5. **Si** no se comunica directamente con el *convocado***,** la *secretaria* registra en el *listado para convocatoria* a quién le informó sobre la *reunión* o el medio de comunicación que utilizó para informarle. La *secretaria* reitera posteriormente la *convocatoria*.

EXCEPCIONES:

TÍTULO: **AVISAR LA CONCURRENCIA**

OBJETIVO:

Registrar la confirmación de la asistencia del *convocado* a la *reunión*.

CONTEXTO:

Ubicación Geográfica: --

Ubicación Temporal: --

Precondición:

Debe realizarse previamente una *convocatoria*, en la cual se solicita a los *convocados* que realicen la *confirmación de asistencia*.

Debe efectuarse con anticipación a la *fecha de la reunión***.**

RECURSOS:

Agenda

listado de convocados

medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:

Convocado

Secretaria

Convocante

EPISODIOS:

1. El *convocado* se comunica por teléfono, fax o personalmente al sitio establecido para la *confirmación de asistencia*, informando al *convocante* o *secretaria* que asistirá a la *reunión*.
2. **Si** el *convocado* no asiste a la *reunión* y asigna un *reemplazante***,** el *convocado* informa los datos del mismo al *convocante* o *secretaria*. RESTRICCIÓN: Debe estar permitido asignar un *reemplazante*.
3. La *secretaria* o el *convocante* registra en la *agenda* la confirmación o los datos del *reemplazante*.
4. La *secretaria* o el *convocante* registra en el *listado de convocados* la confirmación o los datos del *reemplazante*.

EXCEPCIONES:

TITULO: **AVISAR LA NO CONCURRENCIA**

OBJETIVO:
Registrar que el *convocado* no asistirá a la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente una *convocatoria*, en la cual se solicita a los *convocados* que confirmen o no su asistencia.
El *convocado* pudo ya haber informado su concurrencia.
Debe efectuarse con anticipación a la *fecha de la reunión*.

RECURSOS:
Agenda
listado de convocados
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:
Convocado
Secretaria
Convocante

EPISODIOS:
1. El *convocado* se comunica por teléfono, fax o personalmente al sitio establecido para la *confirmación de asistencia*, informando al *convocante* o *secretaria* que no asistirá a la *reunión*. RESTRICCIÓN: el *convocado* no debe exponer ningún *tema*.
2. # La *secretaria* o *convocante* registra en la *agenda* que no asistirá.
3. La *secretaria* o *convocante* registra en el *listado de convocados* que no asistirá.#

EXCEPCIONES:

TÍTULO: **SOLICITAR *MATERIAL FÍSICO***

OBJETIVO:
Reservar el *material físico* que se utilizará en la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe existir una *convocatoria*, en caso de ser el *convocado* quien solicita el *material físico*.
Debe existir la necesidad de una *reunión*, en caso de ser el *convocante* quien necesita el *material físico*.
Debe realizarse con anticipación a la *fecha de la reunión*.

RECURSOS:

Agenda
material físico

ACTORES:
Secretaria
Participante
Convocante

EPISODIOS:
1. El *participante* solicita a la *secretaria* o *convocante* el *material físico* que necesitará para exponer en la *reunión*. RESTRICCIÓN: debe solicitarlo personalmente o por teléfono.
2. El *convocante* o la *secretaria* realiza la reserva del *material físico*.
3. El *convocante* o la *secretaria* registra en la *agenda* el *material físico* reservado.

EXCEPCIONES:
- *Material físico* solicitado no existe. Episodio: 1. (La *secretaria* informa al *participante* la inexistencia del *material físico*).
- *Material físico* solicitado no está disponible para la *fecha establecida*. Episodio: 2. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial).

TÍTULO: **RECORDAR LA *REUNIÓN***

OBJETIVO:
Recordar a los *convocados* la realización de la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente la *convocatoria* a la *reunión*.
El *recordatorio* se realiza a los *convocados* que no dieron *aviso de no concurrencia*.
Debe realizarse antes de la *fecha de la reunión*.

RECURSOS:
listado para convocatoria
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:
Secretaria
Convocados

EPISODIOS:
1. La *secretaria* reitera el aviso, a través de algún medio de comunicación, a cada *convocado* sobre la *fecha establecida*, *horario* y *lugar* de la *reunión*, utilizando el *listado para convocatoria*.

2. **[Si** el *convocado* no ha efectuado el *aviso de concurrencia***,** la *secretaria* pide la *confirmación de asistencia* a la *reunión*.**]**
3. La *secretaria* registra el *recordatorio* al *convocado* en el *listado para convocatoria*.

EXCEPCIONES:

TITULO: **ESTABLECER LA NUEVA *FECHA DE LA REUNIÓN***

OBJETIVO:
Fijar otra fecha y hora de la *reunión*, pudiéndose fijar un nuevo *lugar físico de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe realizarse antes de la *fecha de la reunión*, como consecuencia de un *traslado de fecha*.

RECURSOS:
Agenda
horarios disponibles
espacio físico
material físico

ACTORES:
Convocante
Convocados

EPISODIO:
1. **# Si** no se tienen registrados los *horarios disponibles* de los *convocados***,** SOLICITAR *HORARIOS DISPONIBLES*.
2. El *convocante* consulta la *agenda* para ver sus *horarios disponibles*.
3. El *convocante* consulta sobre la *disponibilidad de espacio*.
4. **[**El *convocante* consulta sobre la disponibilidad del *material físico*.**] #**
5. ESTABLECER LA *FECHA DE LA REUNION*. RESTRICCIÓN: Debe descartar la actual *fecha de la reunión*.

EXCEPCIONES:

Escenarios Excepción

TÍTULO: **ANULAR LA *REUNIÓN***

OBJETIVO:
Liberar la *agenda* de la *reunión* que se cancela.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:
Agenda
listado para convocatoria
cronograma de reuniones
espacio físico
material físico
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:
Convocante
Secretaria
Convocados

EPISODIOS:
1. El *convocante* o la *secretaria* registran la cancelación en la *agenda*.
2. El *convocante* o la *secretaria* registran la cancelación en el *cronograma de reuniones*.
3. **# Si** se realizó la *convocatoria***,** la *secretaria* da el *aviso de cancelación* a cada *convocado* o *reemplazante*, a través de algún medio de comunicación, utilizando el *listado para convocatoria*. RESTRICCIÓN: debe realizarlo el mismo día en que se registra la cancelación.
4. **Si** se realizó la *convocatoria*, la *secretaria* registra el aviso de cancelación en el *listado para convocatoria*.
5. La *secretaria* anula la reserva de *espacio físico*.
6. **[**La *secretaria* anula la reserva de *material físico*.**]** #

EXCEPCIONES:

TÍTULO: **TRASLADAR LA *FECHA DE LA REUNIÓN***

OBJETIVO:
Actualizar la *agenda* por cambio de la *fecha de la reunión* a una nueva fecha.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:

Agenda
listado para convocatoria
cronograma de reuniones
espacio físico
material físico
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:

Convocante
Secretaria
Convocados

EPISODIOS:

1. ESTABLECER LA NUEVA *FECHA DE LA REUNION*.
2. El *convocante* actualiza los datos de la *reunión* en la *agenda*.
3. El *convocante* actualiza los datos de la *reunión* en el *cronograma de reuniones*.
4. **# Si** se realizó la *convocatoria***,** la *secretaria* avisa por algún medio de comunicación el cambio de la *fecha de la reunión* a cada *convocado*, utilizando el *listado para convocatoria*.
5. **Si** se realizó la *convocatoria***,** la *secretaria* registra la comunicación en el *listado para convocatoria*.
6. **Si** hay cambio del *lugar de la reunión***,** la *secretaria* anula la reserva de *espacio físico* y realiza una nueva reserva de *lugar*.
7. **[**La *secretaria* anula la reserva de *material físico* para la fecha anterior y lo reserva para la nueva *fecha de la reunión*.**] #**

EXCEPCIONES:

TÍTULO: **CAMBIAR LOS *REQUERIMIENTOS* DE LA *REUNIÓN***

OBJETIVO:

Actualizar la *agenda* por *cambios en los requerimientos de la reunión*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:

Agenda
listado para convocatoria
espacio físico
material físico

material a presentar
material para repartir
temario
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:
Convocante
Secretaria
Convocados

EPISODIOS:
1. El *convocante* establece un cambio en el *lugar de la reunión*, *temario*, *material a presentar*, *material para repartir* y/o *material físico*.
2. El *convocante* actualiza los datos de la *reunión* en la *agenda*.
3. **# Si** se realizó la *convocatoria***,** la *secretaria* avisa por algún medio de comunicación los *cambios en los requerimientos de la reunión* a cada *convocado*, utilizando el *listado para convocatoria*.
4. **Si** se realizó la *convocatoria***,** la *secretaria* registra la comunicación en el *listado para convocatoria*.
5. **Si** hay cambio del *lugar de la reunión***,** la *secretaria* anula la reserva de *espacio físico* y realiza una nueva reserva de *lugar*.
6. **Si** hay cambio del *lugar de la reunión***,** la *secretaria* anula la reserva de *material físico* para la fecha anterior y lo reserva para la nueva *fecha de la reunión*.
7. **Si** hay cambio del *material físico***,** la *secretaria* anula la reserva de *material físico* y reserva el nuevo *material físico*.
8. **[Si** hay cambio de los *temas* a tratar**,** la *secretaria* envía o entrega el nuevo *temario* a cada *convocado*.**] #**

EXCEPCIONES:

TITULO: **RESERVAR *ESPACIO FISICO* EN EDIFICIO ANEXO**

OBJETIVO:
Definir el *lugar físico de la reunión* según la *fecha establecida* y el *horario*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
No existe *disponibilidad de espacio* en la. *fecha establecida* y el *horario*
Debe obtenerse previamente información de disponibilidad del *material físico*.

RECURSOS:
fecha establecida
horario
material físico disponible

ACTORES:
Convocante

EPISODIO:
1. El *convocante* solicita espacios disponibles en el edificio anexo para la *fecha establecida* y el *horario*.
2. El *convocante* selecciona el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* en el anexo para la *fecha de la reunión* establecida y la disponibilidad de *material físico* requerido.

EXCEPCIONES:
- Conflictos en la *disponibilidad de espacio*. Episodio: 2. (ANULAR LA *REUNIÓN*)
- Conflicto en la disponibilidad de *material físico*. Episodio: 3. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial)

D.4. Ejemplo de Escenarios Futuros

Se han construido dos juegos de escenarios futuros, uno siguiendo un enfoque orientado a lo procedural y otro basado en un enfoque orientado a los objetivos, es decir, el primer juego con pocos cambios en los procesos del negocio y el segundo con un nivel alto en la reingeniería de los procesos.

En la Tabla D-2, se muestra una comparación de cantidad de escenarios según ambos enfoques frente a los escenarios actuales.

Tipos de Escenarios	EA	EF – Baja Reingeniería	EF – Alta Reingeniería
Escenarios Integradores	**1**	**1**	**2**
Escenarios	**4**	**4**	**9**
Sub-Escenarios	**9**	**8**	**9**
Escenarios Excepción	**4**	**4**	**4**
Total de Escenarios	**18**	**17**	**24**
Escenarios con actor Sistema	**--**	**16**	**24**

Tabla D-2. Evolución de EA a EF según enfoques para el Sistema de Agenda de Reuniones

DESCRIPCION DE ESCENARIOS FUTUROS CON BAJA REINGENIERÍA DE LOS PROCESOS DEL NEGOCIO

El escenario integrador permaneció inalterado, se eliminó un sub-escenario donde sus episodios fueron absorbidos por un escenario (operación Aplanar) y se incorporó en todos los escenarios, excepto uno, el actor sistema. Hay un apareo 1 a 1 entre EA y EF.

Objetivos del Sistema:

⇨ Mantener la agenda del convocante
- ✓ Facilitando la selección de la mejor fecha y lugar de cada reunión

Escenario Integrador

TITULO: **SISTEMA DE *AGENDA DE REUNIONES***

OBJETIVO:
Mantener actualizadas los datos de las *reuniones*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe existir un asunto a tratar o resolver por más de una persona..

RECURSOS:

ACTORES:

EPISODIO:
1. REQUERIR UNA *REUNIÓN*.
2. DISEÑAR LA *AGENDA DE REUNIONES*.
3. ORGANIZAR LA *REUNIÓN*.
4. ASISTIR A LA *REUNION*.

EXCEPCIONES:

Escenarios

TÍTULO: **REQUERIR UNA *REUNIÓN***

OBJETIVO:
Determinar el *objetivo* de la *reunión*, los *temas* y los *convocados*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe existir un asunto a tratar o resolver por más de una persona.

RECURSOS:

ACTORES:
Convocante
Secretaria
Sistema

EPISODIOS:
1. El *convocante* define la necesidad de una *reunión*, en base a una demanda externa o propia.
2. El *convocante* determina el *objetivo* de la *reunión*, los *temas* a tratar y los *convocados*.
3. # **[**El *convocante* determina el *material a presentar* por los *convocados*.**]**
4. **[**El *convocante* determina el *material para repartir*.**]** #
5. El *convocante* o la *secretaria* registra en el sistema los datos de la *reunión*: el *objetivo*, los *temas*, los *convocados*, el *material a presentar* y el *material para repartir*, por cada *reunión* a realizarse.

EXCEPCIONES:

TÍTULO: **DISEÑAR LA *AGENDA DE REUNIONES***

OBJETIVO:
Determinar la *fecha de convocatoria*, la *horario*, el *lugar* y *requerimientos* de las *reuniones*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe presentarse previamente la necesidad de una *reunión*.

RECURSOS:
horarios disponibles

ACTORES:
Convocante
Secretaria
Convocados
Sistema

EPISODIOS:
1. El *convocante* solicita los datos de la *reunión* a diseñar al sistema.
2. # **Si** no se tienen registrados los *horarios disponibles* de los *convocados*,

SOLICITAR *HORARIOS DISPONIBLES*.
3. El *convocante* consulta al sistema sobre la *disponibilidad de espacio*.
4. **[**El *convocante* consulta al sistema sobre la disponibilidad de *material físico*.**]**
5. El *convocante* consulta sus *horarios disponibles* en el sistema. #
6. ESTABLECER LA *FECHA DE LA REUNION*.
7. # El *convocante* o la *secretaria* registra en el sistema: *fecha establecida*, *horario* y *lugar*.
8. **[**El *convocante* define el *temario*.**]**
9. **[**El *convocante* o la *secretaria* registra el *temario* en el sistema.**]**
10. Si no se tiene información personal (domicilio, teléfono, e-mail) de algún *convocado***,** la *secretaria* o el *convocante* ingresa al sistema los datos del *convocado*.
11. El *convocante* o la *secretaria* reserva el *espacio físico*.
12. **[**El *convocante* o la *secretaria* reserva el *material físico*.**]**
13. **[**El *convocante* o la *secretaria* registra el *material físico* en el sistema.**]** #

EXCEPCIONES:

TÍTULO: ***ORGANIZAR LA REUNIÓN***

OBJETIVO:
Asegurar el desarrollo eficiente de la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe realizarse previamente el *diseño de la agenda de reuniones*.

RECURSOS:
Material físico
espacio físico

ACTORES:
Convocante
Secretaria
Convocados
Sistema

EPISODIOS:
1. El *convocante* instruye a la *secretaria* sobre la *convocatoria* a la *reunión*, pidiendo los datos al sistema.
2. CONVOCAR A LA *REUNIÓN*.
3. # AVISAR LA CONCURRENCIA.
4. AVISAR LA NO CONCURRENCIA.
5. **[**SOLICITAR *MATERIAL FÍSICO*.**]**
6. **[**RECORDAR LA *REUNIÓN*.**]**
7. La *secretaria* asegura el *espacio físico* para la *fecha de la reunión*. #

EXCEPCIONES:

- Imposibilidad de realizar la *reunión*. (ANULAR LA *REUNIÓN*.)
- Problemas con la *fecha de la reunión*. (TRASLADAR LA *FECHA DE LA REUNIÓN*.)
- Se requiere el cambio de algún *requerimiento* para la *reunión*. (CAMBIAR LOS *REQUERIMIENTOS* DE LA *REUNIÓN*.)

TÍTULO: **ASISTIR A LA *REUNIÓN***

OBJETIVO:

Asistir a la *reunión* según lo establecido en la *convocatoria*.

CONTEXTO:

Ubicación Geográfica: Se realiza en el *lugar físico de la reunión*.

Ubicación Temporal: se realiza en la *fecha establecida* y a la *hora de la reunión*.

Precondición:

El *convocado* o *reemplazante* debe haber confirmado su concurrencia, si fue solicitado.

El *convocado* o *reemplazante* llega al *lugar físico de la reunión* para participar en la misma.

RECURSOS:

listado de convocados
material para repartir
material a presentar
espacio físico

ACTORES:

Secretaria
Convocante
Participante
Sistema

EPISODIOS:

1. El *participante* llega al *lugar de la reunión* en la *fecha establecida* y *horario* establecido en la *convocatoria*.
2. La *secretaria* o el *convocante* registra la asistencia del *participante* en el sistema y en el *listado de convocados*. RESTRICCIÓN: Debe figurar en el *listado de convocados*.
3. # **[**El *participante* lleva el *material para repartir* a todos los *participantes*.**]**
4. **[**El *participante* lleva el *material a presentar*. RESTRICCIÓN: debe ser requerido antes de la *fecha de la reunión*.**]#**
5. La *secretaria* o el *convocante* registra en el sistema la realización de la *reunión*.
6. El sistema elimina los *horarios disponibles* de los *convocados*.

EXCEPCIONES:

- El *participante* no puede llegar a la *reunión*. Episodio: 1. (La *secretaria* o *convocante* registra la ausencia del *participante* en el sistema al finalizar la *reunión*)
- Asiste una persona no invitada en reemplazo de un *convocado*. Episodio: 2.

(El *convocante* determina la participación o no de dicha persona y lo registra en el sistema)

Sub-Escenarios

TÍTULO: **SOLICITAR *HORARIOS DISPONIBLES***

OBJETIVO:
Conocer los *horarios disponibles* de los *convocados*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe presentarse previamente la necesidad de una *reunión*, donde se determinaron los *convocados* a la misma.
No se tienen registrados los *horarios disponibles* de los *convocados* o se desean actualizar.

RECURSOS:
agenda de *convocados*

ACTORES:
Convocado
Secretaria
Convocante
Sistema

EPISODIOS:
1. La *secretaria* o el *convocante* solicita al *convocado* que le informe sus *horarios disponibles*.
2. El *convocado* informa su disponibilidad en base a las fechas y horas libres de su *agenda*.
3. La *secretaria* o el *convocante* registra los *horarios disponibles* en el sistema.

EXCEPCIONES:

TITULO: **ESTABLECER LA *FECHA DE LA REUNIÓN***

OBJETIVO:
Fijar la *fecha de la reunión*, *horario* y *lugar físico de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:

Debe obtenerse previamente información de *horarios disponibles* de los *convocados* y del *convocante*, y la *disponibilidad de espacio* y disponibilidad del *material físico*.

RECURSOS:

horarios disponibles
espacios disponibles
material físico disponible

ACTORES:

Convocante
Sistema

EPISODIO:

1. El *convocante* consulta en el sistema la existencia de fechas comunes disponibles entre los *horarios disponibles* de los *convocados* y los suyos. RESTRICCIÓN: El sistema debe calcular los *horarios disponibles* comunes en menos de 10 segundos el 90% de las veces.
2. **Si** existen fechas comunes disponibles, el *convocante* selecciona la mejor fecha y hora según un criterio dado (la más cercana posible, la más alejada, en horario de oficina, etc.)
3. El *convocante* consulta en el sistema espacios disponibles para la fecha seleccionada y la disponibilidad de *material físico* requerido.
4. El *convocante* selecciona en el sistema el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* para la *fecha de la reunión* establecida y la disponibilidad de *material físico* requerido.

EXCEPCIONES:

- Conflictos en los *horarios disponibles* de los *convocados*. Episodio 2. (ANULAR LA *REUNIÓN*)
- Conflictos en la *disponibilidad de espacio*. Episodio: 4. (RESERVAR *ESPACIO FISICO* EN EDIFICIO ANEXO)
- Conflicto en la disponibilidad de *material físico*. Episodio: 4. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial)

TÍTULO: **CONVOCAR A LA *REUNIÓN***

OBJETIVO:

Invitar a los *convocados* a una *reunión*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:

Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.

RECURSOS:

medios de comunicación (teléfono, fax, correo, computadora, contestador automático, e-mail, etc.)

ACTORES:

Secretaria
Convocados
Sistema

EPISODIOS:

1. La *secretaria* solicita al sistema la impresión del *listado para convocatoria*.
2. La *secretaria* avisa, a través de algún medio de comunicación, a cada *convocado* la *fecha establecida*, *horario* y *lugar* de la *reunión*, utilizando el *listado para convocatoria*.
3. **[**La *secretaria* pide la *confirmación de asistencia* a la *reunión*.**]**
4. **Si** se trata de una *convocatoria* personal**,** solicita al *convocado* que firme la notificación en el *listado para convocatoria*.
5. **[**La *secretaria* solicita al sistema la impresión de copias del *temario* según la cantidad de *convocados*. **]**
6. **[**La *secretaria* entrega o envía una copia del *temario* a cada *convocado***]**.
7. **Si** no se comunica directamente con el *convocado***,** la *secretaria* registra en el *listado para convocatoria* a quién le informó sobre la *reunión* o el medio de comunicación que utilizó para informarle. La *secretaria* reitera posteriormente la *convocatoria*.

EXCEPCIONES:

TÍTULO: **AVISAR LA CONCURRENCIA**

OBJETIVO:

Registrar la confirmación de la asistencia del *convocado* a la *reunión*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:

Debe realizarse previamente una *convocatoria*, en la cual se solicita a los *convocados* que realicen la *confirmación de asistencia*.
Debe efectuarse con anticipación a la *fecha de la reunión***.**

RECURSOS:

listado de convocados
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:

Convocado
Secretaria
Convocante
Sistema

EPISODIOS:

1. El *convocado* se comunica por teléfono, fax o personalmente al sitio establecido para la *confirmación de asistencia*, informando al *convocante* o *secretaria* que asistirá a la *reunión*.
2. **Si** el *convocado* no asiste a la *reunión* y asigna un *reemplazante***,** el *convocado* informa los datos del mismo al *convocante* o *secretaria*. RESTRICCIÓN: Debe estar permitido asignar un *reemplazante*.
3. La *secretaria* o el *convocante* registra en el sistema la confirmación o los datos del *reemplazante*.
4. La *secretaria* o el *convocante* registra en el *listado de convocados* la confirmación o los datos del *reemplazante*.

EXCEPCIONES:

TITULO: **AVISAR LA NO CONCURRENCIA**

OBJETIVO:

Registrar que el *convocado* no asistirá a la *reunión*.

CONTEXTO:

Ubicación Geográfica: --

Ubicación Temporal: --

Precondición:

Debe realizarse previamente una *convocatoria*, en la cual se solicita a los *convocados* que confirmen o no su asistencia.

El *convocado* pudo ya haber informado su concurrencia.

Debe efectuarse con anticipación a la *fecha de la reunión*.

RECURSOS:

listado de convocados

medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:

Convocado

Secretaria

Convocante

Sistema

EPISODIOS:

1. El *convocado* se comunica por teléfono, fax o personalmente al sitio establecido para la *confirmación de asistencia*, informando al *convocante* o *secretaria* que no asistirá a la *reunión*. RESTRICCIÓN: el *convocado* no debe exponer ningún *tema*.
2. # La *secretaria* o *convocante* registra en el sistema que no asistirá.
3. La *secretaria* o *convocante* registra en el *listado de convocados* que no asistirá.#

EXCEPCIONES:

TÍTULO: **SOLICITAR *MATERIAL FÍSICO***

OBJETIVO:
Reservar el *material físico* que se utilizará en la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe existir una *convocatoria*, en caso de ser el *convocado* quien solicita el *material físico*.
Debe existir la necesidad de una *reunión*, en caso de ser el *convocante* quien necesita el *material físico*.
Debe realizarse con anticipación a la *fecha de la reunión*.

RECURSOS:
material físico

ACTORES:
Secretaria
Participante
Convocante
Sistema

EPISODIOS:
1. El *participante* solicita a la *secretaria* o *convocante* el *material físico* que necesitará para exponer en la *reunión*. RESTRICCIÓN: debe solicitarlo personalmente o por teléfono.
2. El *convocante* o la *secretaria* realiza la reserva del *material físico*.
3. El *convocante* o la *secretaria* registra en el sistema el *material físico* reservado.

EXCEPCIONES:
- *Material físico* solicitado no existe. Episodio: 1. (La *secretaria* informa al *participante* la inexistencia del *material físico*).
- *Material físico* solicitado no está disponible para la *fecha establecida*. Episodio: 2. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial).

TÍTULO: **RECORDAR LA *REUNIÓN***

OBJETIVO:
Recordar a los *convocados* la realización de la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente la *convocatoria* a la *reunión*.
El *recordatorio* se realiza a los *convocados* que no dieron *aviso de no*

concurrencia.
Debe realizarse antes de la *fecha de la reunión*.

RECURSOS:
Listado para convocatoria
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:
Secretaria
Convocados

EPISODIOS:
1. La *secretaria* reitera el aviso, a través de algún medio de comunicación, a cada *convocado* sobre la *fecha establecida*, *horario* y *lugar* de la *reunión*, utilizando el *listado para convocatoria*.
2. **[Si** el *convocado* no ha efectuado el *aviso de concurrencia***,** la *secretaria* pide la *confirmación de asistencia* a la *reunión*.**]**
3. La *secretaria* registra el *recordatorio* al *convocado* en el *listado para convocatoria*.

EXCEPCIONES:

TITULO: **ESTABLECER LA NUEVA *FECHA DE LA REUNIÓN***

OBJETIVO:
Fijar otra fecha y hora de la *reunión*, pudiéndose fijar un nuevo *lugar físico de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe realizarse antes de la *fecha de la reunión*, como consecuencia de un *traslado de fecha*.

RECURSOS:
horarios disponibles
espacio físico
material físico

ACTORES:
Convocante
Convocados
Sistema

EPISODIO:
1. **# Si** no se tienen registrados los *horarios disponibles* de los *convocados***,** SOLICITAR *HORARIOS DISPONIBLES*.
2. El *convocante* consulta al sistema para ver sus *horarios disponibles*.
3. El *convocante* consulta al sistema sobre la *disponibilidad de espacio*.

4. **[**El *convocante* consulta al sistema sobre la disponibilidad del *material físico*.**]** #
5. ESTABLECER LA *FECHA DE LA REUNION*. RESTRICCIÓN: El sistema debe descartar la actual *fecha de la reunión*.

EXCEPCIONES:

Escenarios Excepción

TÍTULO: **ANULAR LA *REUNIÓN***

OBJETIVO:

Liberar la *agenda* de la *reunión* que se cancela.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:

listado para convocatoria
espacio físico
material físico
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:

Convocante
Secretaria
Convocados
Sistema

EPISODIOS:

1. El *convocante* o la *secretaria* registran la cancelación en el sistema.
2. **# Si** se realizó la *convocatoria***,** la *secretaria* da el *aviso de cancelación* a cada *convocado* o *reemplazante*, a través de algún medio de comunicación, utilizando el *listado para convocatoria*. RESTRICCIÓN: debe realizarlo el mismo día en que se registra la cancelación.
3. **Si** se realizó la *convocatoria*, la *secretaria* registra el aviso de cancelación en el *listado para convocatoria*.
4. La *secretaria* anula la reserva de *espacio físico*.
5. **[**La *secretaria* anula la reserva de *material físico*.**]** #

EXCEPCIONES:

TÍTULO: **TRASLADAR LA *FECHA DE LA REUNIÓN***

OBJETIVO:
Actualizar la *agenda* por cambio de la *fecha de la reunión* a una nueva fecha.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:
listado para convocatoria
espacio físico
material físico
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:
Convocante
Secretaria
Convocados
Sistema

EPISODIOS:
1. ESTABLECER LA NUEVA *FECHA DE LA REUNION*.
2. El *convocante* actualiza los datos de la *reunión* en el sistema.
3. **# Si** se realizó la *convocatoria***,** la *secretaria* avisa por algún medio de comunicación el cambio de la *fecha de la reunión* a cada *convocado*, utilizando el *listado para convocatoria*.
4. **Si** se realizó la *convocatoria***,** la *secretaria* registra la comunicación en el *listado para convocatoria*.
5. **Si** hay cambio del *lugar de la reunión***,** la *secretaria* anula la reserva de *espacio físico* y realiza una nueva reserva de *lugar*.
6. **[**La *secretaria* anula la reserva de *material físico* para la fecha anterior y lo reserva para la nueva *fecha de la reunión*.**] #**

EXCEPCIONES:

TÍTULO: **CAMBIAR LOS *REQUERIMIENTOS* DE LA *REUNIÓN***

OBJETIVO:
Actualizar la *agenda* por *cambios en los requerimientos de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:

Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:

listado para convocatoria
espacio físico
material físico
material a presentar
material para repartir
medios de comunicación (teléfono, fax, correo, computadora, etc.)

ACTORES:

Convocante
Secretaria
Convocados
Sistema

EPISODIOS:

1. El *convocante* establece un cambio en el *lugar de la reunión*, *temario*, *material a presentar*, *material para repartir* y/o *material físico*.
2. El *convocante* actualiza los datos de la *reunión* en el sistema.
3. **# Si** se realizó la *convocatoria***,** la *secretaria* avisa por algún medio de comunicación los *cambios en los requerimientos de la reunión* a cada *convocado*, utilizando el *listado para convocatoria*.
4. **Si** se realizó la *convocatoria***,** la *secretaria* registra la comunicación en el *listado para convocatoria*.
5. **Si** hay cambio del *lugar de la reunión***,** la *secretaria* anula la reserva de *espacio físico* y realiza una nueva reserva de *lugar*.
6. **Si** hay cambio del *lugar de la reunión***,** la *secretaria* anula la reserva de *material físico* para la fecha anterior y lo reserva para la nueva *fecha de la reunión*.
7. **Si** hay cambio del *material físico***,** la *secretaria* anula la reserva de *material físico* y reserva el nuevo *material físico*.
8. **[Si** hay cambio de los *temas* a tratar**,** la *secretaria* solicita al sistema la impresión del nuevo *temario* para cada *convocado*.**]**
9. **[Si** hay cambio de los *temas* a tratar**,** la *secretaria* envía o entrega el nuevo *temario* a cada *convocado*.**] #**

EXCEPCIONES:

TITULO: **RESERVAR *ESPACIO FISICO* EN EDIFICIO ANEXO**

OBJETIVO:

Definir el *lugar físico de la reunión* según la *fecha establecida* y el *horario*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
No existe *disponibilidad de espacio* en la. *fecha establecida* y el *horario*

Debe obtenerse previamente información de disponibilidad del *material físico*.

RECURSOS:
fecha establecida
horario
material físico disponible

ACTORES:
Convocante
Sistema

EPISODIO:
1. El *convocante* solicita espacios disponibles en el edificio anexo para la *fecha establecida* y el *horario*.
2. El *convocante* selecciona el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* en el anexo para la *fecha de la reunión* establecida y la disponibilidad de *material físico* requerido.
3. El *convocante* registra en el sistema el *lugar físico de la reunión*.

EXCEPCIONES:
- Conflictos en la *disponibilidad de espacio*. Episodio: 2. (ANULAR LA *REUNIÓN*)
- Conflicto en la disponibilidad de *material físico*. Episodio: 3. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial)

DESCRIPCION DE ESCENARIOS FUTUROS CON ALTA REINGENIERÍA DE LOS PROCESOS DEL NEGOCIO

El escenario integrador considera en uno de sus episodios al mismo escenario integrador actual que antes era de primer nivel. Todos los escenarios involucran al actor sistema. Sobre 24 EF, hay 14 escenarios que aparean con escenarios actuales, pero cuyos episodios han sufrido un cambio importante. Respecto a los EA, ocurrió un aplanamiento de un sub-escenario y la consolidación de 2 EA en un EF, posteriormente este EF fue dividido para tratar por separado convocados pertenecientes a la organización y convocados externos.

Objetivos del Sistema:
⇨ Mantener una agenda integrada del personal de la organización
- ✓ Facilitando la selección de la mejor fecha y lugar de cada reunión
- ✓ Asegurando la mayor concurrencia de convocados a la reunión
- ✓ Facilitando el manejo de convocados no pertenecientes a la organización.

Escenarios Integradores

TITULO: **SISTEMA DE *AGENDA DE REUNIONES***

OBJETIVO:

Mantener *agendas de reuniones* interconectadas para el personal de la organización.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: El personal tiene asignado un nombre de usuario y clave.

RECURSOS:

ACTORES:

EPISODIO:

1. #ABRIR UNA *AGENDA DE REUNIONES*.
2. CARGAR *CONVOCADOS* EXTERNOS.
5. CARGAR ESPACIOS DISPONIBLES.
6. CARGAR *MATERIAL FÍSICO* DISPONIBLE.
3. MANTENER LA *AGENDA DE REUNIONES*.
4. CONSULTAR EL *CRONOGRAMA DE REUNIONES*. #

EXCEPCIONES:

TITULO: **MANTENER LA *AGENDA DE REUNIONES***

OBJETIVO:

Mantener actualizadas los datos de las *reuniones*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe existir un asunto a tratar o resolver por más de una persona..

RECURSOS:

ACTORES:

EPISODIO:

1. REQUERIR UNA *REUNIÓN*.
2. DISEÑAR LA *AGENDA DE REUNIONES*.
3. ORGANIZAR LA *REUNIÓN*.
4. ASISTIR A LA *REUNION*.

EXCEPCIONES:

Escenarios

TITULO: **ABRIR UNA *AGENDA DE REUNIONES***

OBJETIVO:
Generar una *agenda* para un empleado de la organización.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
El empleado debe estar autorizado a tener una *agenda de reuniones*.

RECURSOS:
Nombre de usuario y clave
Datos del empleado

ACTORES:
Empleado
Sistema

EPISODIO:
1. El empleado ingresa al sistema un nombre de usuario y clave.
2. El sistema solicita datos del empleado: nombre y apellido, e-mail, teléfono, cargo y sector.
3. El sistema genera una *agenda de reuniones* para dicho empleado con los datos ingresados.

EXCEPCIONES:
- Nombre de usuario inexistente o clave inválida. Episodio 1. (El sistema envía un mensaje de error y reitera el pedido 3 veces).
- Usuario ya tiene una *agenda* abierta. Episodio 1. (El sistema envía un mensaje informativo y termina).
- Algún dato del empleado es inválido o nulo. Episodio 2. (El sistema envía mensaje de error y reitera una nueva carga de datos)

TITULO: **CARGAR *CONVOCADOS* EXTERNOS**

OBJETIVO:
Ingresar datos de posibles *convocados* a una *reunión* externos a la organización.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --

Precondición:
Un posible *convocado* a una *reunión* no pertenece a la organización.

RECURSOS:
Nombre de usuario y clave
Datos del *convocado*

ACTORES:
Convocante
Sistema

EPISODIO:
1. El *convocante* ingresa al sistema su nombre de usuario y clave.
2. El sistema solicita datos del *convocado* externo: nombre y apellido, e-mail, teléfono, empresa, cargo y sector.
3. El sistema registra los datos del *convocado* externo.

EXCEPCIONES:
- Nombre de usuario inexistente o clave inválida. Episodio 1. (El sistema envía un mensaje de error y reitera el pedido 3 veces).
- Algún dato del *convocado* es inválido o nulo o *convocado* ya existe o tiene una *agenda*. Episodio 2. (El sistema envía mensaje de error y reitera una nueva carga de datos)

TITULO: **CARGAR ESPACIOS DISPONIBLES**

OBJETIVO:
Mantener en el sistema los espacios disponibles para *reuniones*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
El espacio puede estar en el edificio de la organización o en el edificio anexo.

RECURSOS:
Nombre de usuario y clave
Datos del espacio

ACTORES:
Empleado
Sistema

EPISODIO:
1. El empleado ingresa al sistema un nombre de usuario y clave.
2. El sistema solicita datos del espacio: identificación, ubicación, capacidad de personas y si es anexo o no.
3. El sistema registra los datos del espacio.

EXCEPCIONES:

- Nombre de usuario inexistente o clave inválida. Episodio 1. (El sistema envía un mensaje de error y reitera el pedido 3 veces).
- Algún dato del espacio es inválido o nulo o espacio ya existe. Episodio 2. (El sistema envía mensaje de error y reitera una nueva carga de datos)

TITULO: **CARGAR *MATERIAL FÍSICO* DISPONIBLE**

OBJETIVO:

Mantener en el sistema los *materiales físicos* disponibles para *reuniones*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
El *material físico* está disponible para *reuniones*.

RECURSOS:

Nombre de usuario y clave
Datos del *material físico*

ACTORES:

Empleado
Sistema

EPISODIO:

1. El empleado ingresa al sistema un nombre de usuario y clave.
2. El sistema solicita datos del *material físico*: identificación, nombre y descripción
3. El sistema registra los datos del *material físico*.

EXCEPCIONES:

- Nombre de usuario inexistente o clave inválida. Episodio 1. (El sistema envía un mensaje de error y reitera el pedido 3 veces).
- Algún dato del *material físico* es inválido o nulo o *material físico* ya existe. Episodio 2. (El sistema envía mensaje de error y reitera una nueva carga de datos)

TITULO: **CONSULTAR EL *CRONOGRAMA DE REUNIONES***

OBJETIVO:

Informarse sobre las reuniones diseñadas en un período de tiempo dado.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe haberse realizado previamente el *diseño de la agenda de reuniones*.

RECURSOS:
Período a consultar

ACTORES:
Convocante
Sistema

EPISODIO:
1. El *convocante* solicita al sistema el *cronograma de reuniones* para un período dado.
2. El sistema busca *reuniones* con *fecha establecida* en el período estipulado y arma el *cronograma de reuniones*..
3. El sistema muestra el *cronograma de reuniones*. RESTRICCION: El sistema debe manipular colores para la visualización del *cronograma de reuniones*. RESTRICCION: El sistema debe mostrar el *cronograma de reuniones* en menos de 30 segundos el 100% de las veces.

EXCEPCIONES:

TÍTULO: **REQUERIR UNA *REUNIÓN***

OBJETIVO:
Determinar el *objetivo* de la *reunión*, los *temas* y los *convocados*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe existir un asunto a tratar o resolver por más de una persona.

RECURSOS:

ACTORES:
Convocante
Secretaria
Sistema

EPISODIOS:
1. El *convocante* define la necesidad de una *reunión*, en base a una demanda externa o propia.
2. El *convocante* determina el *objetivo* de la *reunión*, los *temas* a tratar y los *convocados*. RESTRICCIÓN: El *convocado* debe tener una *agenda* o ser un *convocado* externo.
3. **# [**El *convocante* determina el *material a presentar* por los *convocados*.**]**
4. **[**El *convocante* determina el *material para repartir*.**] #**
5. **[Si** el *convocante* será un *participante*, el *convocante* determina el *material fisico*.**] #**
6. El *convocante* o la *secretaria* registra en el sistema los datos de la *reunión*: el *objetivo*, los *temas*, los *convocados*, el *material a presentar*, el *material para*

repartir y el *material físico.*
7. El *convocante* registra en el sistema si va a ser un *participante* de la *reunión*.
8. El *convocante* registra en el sistema si se requiere *confirmación de asistencia* a la *reunión*.

EXCEPCIONES:

TÍTULO: **DISEÑAR LA *AGENDA DE REUNIONES***

OBJETIVO:
Determinar la *fecha de convocatoria*, el *horario*, el *lugar* y los *requerimientos* de las *reuniones*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe presentarse previamente la necesidad de una *reunión*.

RECURSOS:
horarios disponibles
espacios disponibles
material físico disponible

ACTORES:
Convocante
Secretaria
Convocados
Sistema

EPISODIOS:
1. El *convocante* solicita los datos de la *reunión* a diseñar al sistema.
2. # El sistema INFORMA *HORARIOS DISPONIBLES*.
3. **Si** hay *convocados* externos **entonces** SOLICITAR *HORARIOS DISPONIBLES* A *CONVOCADOS* EXTERNOS.
4. El *convocante* consulta al sistema sobre la *disponibilidad de espacio*.
   RESTRICCION: Debe elegir el espacio según la cantidad de *participantes* .
5. **Si** se requiere *material físico* **entonces** el *convocante* consulta al sistema sobre la disponibilidad de *material físico*. #
6. ESTABLECER LA *FECHA DE LA REUNION*.
7. # El *convocante* o la *secretaria* registra en el sistema: *fecha establecida*, *horario* y *lugar*.
8. **[**El *convocante* define el *temario*.**]**
9. **[**El *convocante* o la *secretaria* registra el *temario* en el sistema.**]**
10. El *convocante* o la *secretaria* reserva el *espacio físico*.
11. **[**El *convocante* o la *secretaria* reserva el *material físico*.**]**#

EXCEPCIONES:

TÍTULO: ***ORGANIZAR LA REUNIÓN***

OBJETIVO:
Asegurar el desarrollo eficiente de la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición: Debe realizarse previamente el *diseño de la agenda de reuniones*.

RECURSOS:
material físico
espacio físico

ACTORES:
Convocante
Secretaria
Convocados
Sistema

EPISODIOS:
1. El *convocante* o la *secretaria* realizan en el sistema CONVOCAR A LA *REUNIÓN*.
2. # AVISAR LA ASISTENCIA.
3. AVISAR LA ASISTENCIA DE *CONVOCADO* EXTERNO.
4. **[**SOLICITAR *MATERIAL FÍSICO*.**]**
5. **[**RECORDAR LA *REUNIÓN*.**]** #

EXCEPCIONES:
- Imposibilidad de realizar la *reunión*. (ANULAR LA *REUNIÓN*.)
- Problemas con la *fecha de la reunión*. (TRASLADAR LA *FECHA DE LA REUNIÓN*.)
- Se requiere el cambio de algún *requerimiento* para la *reunión*. (CAMBIAR LOS *REQUERIMIENTOS* DE LA *REUNIÓN*.)

TÍTULO: **ASISTIR A LA *REUNIÓN***

OBJETIVO:
Asistir a la *reunión* según lo establecido en la *convocatoria*.

CONTEXTO:
Ubicación Geográfica: Se realiza en el *lugar físico de la reunión*.
Ubicación Temporal: se realiza en la *fecha establecida* y a la *hora de la reunión*.
Precondición:
El *convocado* o *reemplazante* debe haber confirmado su concurrencia, si fue solicitado.
El *participante* llega al *lugar físico de la reunión* para participar en la misma.

RECURSOS:

material para repartir
material a presentar
espacio físico

ACTORES:

Secretaria
Convocante
Participante
Sistema

EPISODIOS:

1. El *participante* llega al *lugar de la reunión* en la *fecha establecida* y *horario* establecido en la *convocatoria*.
2. La *secretaria* o el *convocante* registra la asistencia del *participante* en el sistema.
3. # **[**El *participante* lleva el *material para repartir* a todos los *participantes*.**]**
4. **[**El *participante* lleva el *material a presentar*. RESTRICCIÓN: debe ser requerido antes de la *fecha de la reunión*.**]#**
5. La *secretaria* o el *convocante* registra en el sistema la realización de la *reunión*, liberando el *espacio físico* y el *material físico*.
6. El sistema elimina los *horarios disponibles* de los *convocados* externos.

EXCEPCIONES:

- El *participante* no puede llegar a la *reunión*. Episodio: 1. (La *secretaria* o *convocante* registra la ausencia del *participante* en el sistema al finalizar la *reunión*)
- Asiste una persona no invitada en reemplazo de un *convocado*. Episodio: 2. (El *convocante* determina la participación o no de dicha persona y lo registra en el sistema)

Sub-Escenarios

TÍTULO: **SOLICITAR *HORARIOS DISPONIBLES* A *CONVOCADOS* EXTERNOS**

OBJETIVO:

Conocer los *horarios disponibles* de los *convocados* externos.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:

Debe presentarse previamente la necesidad de una *reunión*, donde existen *convocados* externos a la misma.
No se tienen registrados los *horarios disponibles* de los *convocados* externos o se desean actualizar.

RECURSOS:

ACTORES:
Convocado
Secretaria
Convocante
Sistema

EPISODIOS:
1. La *secretaria* o el *convocante* solicita al *convocado* que le informe sus *horarios disponibles*.
2. El *convocado* informa su disponibilidad en base a sus fechas y horas libres.
3. La *secretaria* o el *convocante* registra los *horarios disponibles* en el sistema.

EXCEPCIONES:
- El *convocado* externo no tiene *horarios disponibles*. Episodio 2. (El *convocante* elimina del sistema al *convocado* externo para esa *reunión*)

TÍTULO: **INFORMAR *HORARIOS DISPONIBLES***

OBJETIVO:
Conocer los *horarios disponibles* de los *convocados*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe presentarse previamente la necesidad de una *reunión*, donde se determinaron los *convocados* a la misma.
Los *convocados* pertenecen a la organización.

RECURSOS:
Agenda de *convocados*

ACTORES:
Sistema

EPISODIOS:
1. El sistema obtiene los *horarios disponibles* de la *agenda* cada *convocado*.
2. El sistema determina los *horarios disponibles* comunes a todos los *convocados*.
3. El sistema informa dichos *horarios disponibles* comunes. RESTRICCIÓN: El sistema debe calcular los *horarios disponibles* comunes en menos de 10 segundos el 90% de las veces.

EXCEPCIONES:

TITULO: **ESTABLECER LA *FECHA DE LA REUNIÓN***

OBJETIVO:

Fijar la *fecha de la reunión*, *horario* y *lugar físico de la reunión*.

CONTEXTO:

Ubicación Geográfica: --

Ubicación Temporal: --

Precondición:

Debe obtenerse previamente información de *horarios disponibles* de los *participantes*, y la *disponibilidad de espacio* y disponibilidad del *material físico*.

RECURSOS:

Horarios disponibles

Espacios disponibles

Material físico disponible

ACTORES:

Convocante

Sistema

EPISODIO:

1. El sistema muestra las fechas y horas comunes disponibles entre todos los *participantes*. RESTRICCIÓN: El sistema debe calcular los *horarios disponibles* comunes en menos de 10 segundos el 90% de las veces.
2. **Si** existen fechas comunes disponibles, el *convocante* selecciona la mejor fecha y hora según un criterio dado (la más cercana posible, la más alejada, en horario de oficina, etc.)
3. El *convocante* consulta en el sistema espacios disponibles para la fecha seleccionada y la disponibilidad de *material físico* requerido.
4. El *convocante* selecciona en el sistema el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* para la *fecha de la reunión* establecida y la disponibilidad de *material físico* requerido.

EXCEPCIONES:

- Conflictos en los *horarios disponibles* de los *participantes*. Episodio 2. (ANULAR LA *REUNIÓN*)
- Conflictos en la *disponibilidad de espacio*. Episodio: 4. (RESERVAR *ESPACIO FISICO* EN EDIFICIO ANEXO)
- Conflicto en la disponibilidad de *material físico*. Episodio: 4. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial)

TÍTULO: **CONVOCAR A LA *REUNIÓN***

OBJETIVO:

Invitar a los *convocados* a una *reunión*.

CONTEXTO:

Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse 5 días antes de la *fecha de la reunión*.

RECURSOS:
Medio de comunicación (e-mail de *convocados*)
Medio de comunicación de respuesta (e-mail de *secretaria* y *convocante*)
fecha establecida, *horario*, *lugar* de la *reunión*
temario
listado de convocados
frecuencia para el *recordatorio*

ACTORES:
Secretaria
Sistema

EPISODIOS:
1. La *secretaria* solicita al sistema que realice la *convocatoria*.
2. El sistema envía a través de un medio de comunicación, a cada *convocado* la *fecha establecida*, *horario*, *lugar* de la *reunión*, el *temario* y el medio de comunicación de respuesta.
3. Si se requiere *confirmación de asistencia*, el sistema pide la *confirmación de asistencia* a la *reunión* a cada *convocado*.
4. El sistema registra el envío de la *convocatoria* en el *listado de convocados*.
5. La *secretaria* registra en el sistema la frecuencia en la que debe realizarse el *recordatorio* para la *reunión*.

EXCEPCIONES:

TÍTULO: **AVISAR LA ASISTENCIA**

OBJETIVO:
Registrar la confirmación de la asistencia o no del *convocado* a la *reunión*, perteneciendo éste a la organización.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente una *convocatoria*, en la cual se solicita a los *convocados* que realicen la *confirmación de asistencia*.
Debe efectuarse con anticipación a la *fecha de la reunión***.**
El *convocado* pudo ya haber informado su concurrencia y ahora informa su ausencia o reemplazo.
El *convocado* debe pertenecer a la organización.

RECURSOS:

listado de convocados

ACTORES:
convocado
sistema

EPISODIOS:
1. El *convocado* ingresa en el sistema si asiste o no a la *reunión*. RESTRICCIÓN: el *convocado* no debe exponer ningún *tema* si informa su ausencia.
2. **Si** el *convocado* no asiste a la *reunión* y asigna un *reemplazante***,** el *convocado* informa al sistema los datos del mismo. RESTRICCIÓN: Debe estar permitido asignar un *reemplazante*. RESTRICCIÓN: El *reemplazante* debe tener abierta una *agenda de reuniones*.
3. **Si** el *convocado* asiste**,** el sistema registra la *reunión* en la *agenda* del *convocado*.
4. **Si** el *convocado* no asiste o envía un *reemplazante* y ya había avisado su asistencia**,** el sistema registra la baja de la *reunión* en la *agenda* del *convocado*.
5. El sistema registra en el *listado de convocados* la confirmación, ausencia o los datos del *reemplazante*.

EXCEPCIONES:

TÍTULO: **AVISAR LA ASISTENCIA DE *CONVOCADO* EXTERNO**

OBJETIVO:
Registrar la confirmación de la asistencia o no del *convocado* externo a la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente una *convocatoria*, en la cual se solicita a los *convocados* que realicen la *confirmación de asistencia*.
Debe efectuarse con anticipación a la *fecha de la reunión***.**
El *convocado* pudo ya haber informado su concurrencia y ahora informa su ausencia o reemplazo.
El *convocado* debe ser externo a la organización.

RECURSOS:
listado de convocados
medios de comunicación (e-mail.)

ACTORES:
convocado
secretaria
convocante
sistema

EPISODIOS:

1. El *convocado* envía a través de un medio de comunicación la *confirmación de asistencia*, informando que asistirá o no a la *reunión*. RESTRICCIÓN: el *convocado* no debe exponer ningún *tema* si informa su ausencia.
2. **Si** el *convocado* no asiste a la *reunión* y asigna un *reemplazante***,** el *convocado* informa los datos del mismo. RESTRICCIÓN: Debe estar permitido asignar un *reemplazante*.
3. **Si** se asignó un *reemplazante***,** el *convocante* CARGA *CONVOCADOS* EXTERNOS.
4. La *secretaria* o el *convocante* registra en el sistema la confirmación, ausencia o el reemplazo.
5. El sistema registra en el *listado de convocados* la confirmación, ausencia o los datos del *reemplazante*.

EXCEPCIONES:

TÍTULO: **SOLICITAR *MATERIAL FÍSICO***

OBJETIVO:

Reservar el *material físico* que se utilizará en la *reunión*.

CONTEXTO:

Ubicación Geográfica: --

Ubicación Temporal: --

Precondición:

Debe existir una *convocatoria*, en caso de ser el *convocado* quien solicita el *material físico*.

Debe existir la necesidad de una *reunión*, en caso de ser el *convocante* quien necesita el *material físico*.

Debe realizarse con anticipación a la *fecha de la reunión*.

RECURSOS:

material físico

medio de comunicación (e-mail)

ACTORES:

Secretaria

Participante

Convocante

Sistema

EPISODIOS:

1. El *participante* envía a través de un medio de comunicación su solicitud de *material físico* que necesitará para exponer en la *reunión*.
2. El *convocante* o la *secretaria* registra en el sistema la reserva del *material físico*.

EXCEPCIONES:

- *Material físico* solicitado no existe. Episodio: 1. (La *secretaria* informa al *participante* la inexistencia del *material físico*).

- *Material físico* solicitado no está disponible para la *fecha establecida*. Episodio: 2. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial).

TÍTULO: **RECORDAR LA *REUNIÓN***

OBJETIVO:
Recordar a los *convocados* la realización de la *reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente la *convocatoria* a la *reunión*.
El *recordatorio* se realiza a los *convocados* que no dieron *aviso de no concurrencia*.
Debe realizarse antes de la *fecha de la reunión*.

RECURSOS:
listado para convocatoria
medios de comunicación (e-mail)
frecuencia de *recordatorio*

ACTORES:
Sistema

EPISODIOS:
1. El sistema envía por algún medio de comunicación, a cada *convocado* sobre la *fecha establecida*, *horario* y *lugar* de la *reunión*, utilizando el *listado para convocatoria* y según la frecuencia de *recordatorio*.
2. El sistema verifica en el *listado de convocados* aquellos *convocados* que no avisaron sobre su asistencia y cuyo aviso es requerido.
3. **Si** el *convocado* no ha efectuado el *aviso de concurrencia* y se requiere *confirmación de asistencia*, el sistema solicita por el mismo medio de comunicación la *confirmación de asistencia* a la *reunión*.
4. El sistema registra el *recordatorio* al *convocado* en el *listado para convocatoria*.

EXCEPCIONES:

TITULO: **ESTABLECER LA NUEVA *FECHA DE LA REUNIÓN***

OBJETIVO:
Fijar otra fecha y hora de la *reunión*, pudiéndose fijar un nuevo *lugar físico de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --

Precondición:
Debe realizarse antes de la *fecha de la reunión*, como consecuencia de un *traslado de fecha*.

RECURSOS:
horarios disponibles
espacio físico
material físico

ACTORES:
Convocante
Convocados
Sistema

EPISODIO:
1. El *convocante* ingresa al sistema una nueva posible *fecha de la reunión*.
2. **# Si** no se tienen registrados los *horarios disponibles* de los *convocados* externos**,** SOLICITAR *HORARIOS DISPONIBLES* A *CONVOCADOS* EXTERNOS.
3. El sistema INFORMA *HORARIOS DISPONIBLES*.
4. El *convocante* consulta al sistema sobre la *disponibilidad de espacio*.
5. **[**El *convocante* consulta al sistema sobre la disponibilidad del *material físico*.**] #**
6. ESTABLECER LA *FECHA DE LA REUNION*. RESTRICCIÓN: El sistema debe descartar la actual *fecha de la reunión*.

EXCEPCIONES:

Escenarios Excepción

TÍTULO: **ANULAR LA *REUNIÓN***

OBJETIVO:
Liberar la *agenda* de la *reunión* que se cancela.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:
listado para convocatoria
espacio físico
material físico
medios de comunicación (e-mail)

ACTORES:
Convocante
Secretaria
Sistema

EPISODIOS:
1. El *convocante* o la *secretaria* registra la cancelación en el sistema.
2. **# Si** se realizó la *convocatoria***,** el sistema envía por algún medio de comunicación el *aviso de cancelación* a cada *convocado* o *reemplazante*, utilizando el *listado para convocatoria*. RESTRICCIÓN: debe realizarlo el mismo día en que se registra la cancelación.
3. **Si** se realizó la *convocatoria*, el sistema registra el *aviso de cancelación* en el *listado para convocatoria*.
4. El sistema anula la reserva de *espacio físico*.
5. **[**El sistema anula la reserva de *material físico*.**]** **#**

EXCEPCIONES:

TÍTULO: **TRASLADAR LA *FECHA DE LA REUNIÓN***

OBJETIVO:
Actualizar la *agenda* por cambio de la *fecha de la reunión* a una nueva fecha.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse mínimo 2 días antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:
listado para convocatoria
espacio físico
material físico
medios de comunicación (e-mail)

ACTORES:
Convocante
Sistema

EPISODIOS:
1. ESTABLECER LA NUEVA *FECHA DE LA REUNION*.
2. El *convocante* actualiza los datos de la *reunión* en el sistema.
3. **# Si** se realizó la *convocatoria***,** el sistema avisa por algún medio de comunicación el cambio de la *fecha de la reunión* a cada *convocado*, utilizando el *listado para convocatoria*.
4. **Si** se realizó la *convocatoria***,** el sistema registra la comunicación en el *listado*

para convocatoria.
5. **Si** hay cambio del *lugar de la reunión***,** el sistema anula la reserva de *espacio físico* y registra una nueva reserva de *lugar*.
6. **[**El sistema anula la reserva de *material físico* para la fecha anterior y lo reserva para la nueva *fecha de la reunión*.**]** #

EXCEPCIONES:

TÍTULO: **CAMBIAR LOS *REQUERIMIENTOS* DE LA *REUNIÓN***

OBJETIVO:
Actualizar la *agenda* por *cambios en los requerimientos de la reunión*.

CONTEXTO:
Ubicación Geográfica: --
Ubicación Temporal: --
Precondición:
Debe realizarse previamente el *diseño de la agenda de reuniones*.
Debe realizarse antes de la *fecha de la reunión*.
Pudo o no haberse realizado la *convocatoria*.

RECURSOS:
listado para convocatoria
espacio físico
material físico
material a presentar
material para repartir
medios de comunicación (e-mail)

ACTORES:
Convocante
Secretaria
Convocados
Sistema

EPISODIOS:
1. El *convocante* establece un cambio en el *lugar de la reunión*, *temario*, *material a presentar*, *material para repartir* y/o *material físico*.
2. El *convocante* actualiza los datos de la *reunión* en el sistema.
3. # **Si** se realizó la *convocatoria***,** el sistema avisa por algún medio de comunicación los *cambios en los requerimientos de la reunión* a cada *convocado*, utilizando el *listado para convocatoria*.
4. **Si** se realizó la *convocatoria***,** el sistema registra la comunicación en el *listado para convocatoria*.
5. **Si** hay cambio del *lugar de la reunión***,** el sistema anula la reserva de *espacio físico* y realiza una nueva reserva de *lugar*.
6. **[Si** hay cambio del *lugar de la reunión***,** el sistema anula la reserva de *material físico* para la fecha anterior y lo reserva para la nueva *fecha de la reunión*.**]**
7. **Si** hay cambio del *material físico***,** el sistema anula la reserva de *material físico* y

reserva el nuevo *material físico*.

8. **Si** hay cambio de los *temas* a tratar**,** el sistema envía por algún medio de comunicación el nuevo *temario* a cada *convocado*. #

EXCEPCIONES:

TITULO: **RESERVAR *ESPACIO FISICO* EN EDIFICIO ANEXO**

OBJETIVO:

Definir el *lugar físico de la reunión* según la *fecha establecida* y el *horario*.

CONTEXTO:

Ubicación Geográfica: --

Ubicación Temporal: --

Precondición:

No existe *disponibilidad de espacio* en la. *fecha establecida* y el *horario*
Debe obtenerse previamente información de disponibilidad del *material físico*.

RECURSOS:

fecha establecida
horario
material físico disponible

ACTORES:

Convocante
Sistema

EPISODIO:

1. El *convocante* solicita al sistema espacios disponibles en el edificio anexo para la *fecha establecida* y el *horario*.
2. El *convocante* selecciona el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* en el anexo para la *fecha de la reunión* establecida y la disponibilidad de *material físico* requerido.
3. El *convocante* registra en el sistema el *lugar físico de la reunión*.

EXCEPCIONES:

- Conflictos en la *disponibilidad de espacio*. Episodio: 2. (ANULAR LA *REUNIÓN*)
- Conflicto en la disponibilidad de *material físico*. Episodio: 3. (El *convocante* solicita autorización al Sector Mantenimiento para obtener *material físico* especial)

D.5. Ejemplo de Lista de Requisitos

A continuación se muestran dos listas de requisitos, una obtenida a partir de los EF construidos bajo un enfoque procedural y la segunda lista basada en EF construidos con un enfoque dirigido por objetivos. Esta lista distingue los RF de los RNF. Podría aplicarse otra taxonomía de requisitos (ver sub-sección 2.6.4) o utilizar por ejemplo el estándar [IEEE Std 830-1998] con ocho propuestas en sus anexos.

LISTA DE REQUISITOS CON BAJA REINGENIERÍA DE LOS PROCESOS DEL NEGOCIO

REQUISITOS FUNCIONALES
✓ El sistema debe mantener los datos de la *reunión*: el *objetivo*, los *temas*, los *convocados*, el *material a presentar* y el *material para repartir*, por cada *reunión* a realizarse.
✓ El sistema debe mostrar los datos de la *reunión* a pedido del *convocante* o *secretaria*.
✓ El sistema debe mostrar la *disponibilidad de espacio*.
✓ El sistema debe mostrar la disponibilidad de *material físico*.
✓ El sistema debe mostrar los *horarios disponibles* del *convocante*.
✓ El sistema debe mantener la *fecha establecida*, *horario*, *lugar*, *temario* (opcional) y el *material físico* (opcional) por cada *reunión* diseñada.
✓ El sistema mantiene datos de los *convocados* (nombre y apellido, domicilio, teléfono, e-mail)
✓ El sistema debe mantener la asistencia o ausencia de los *participantes* a cada *reunión*.
✓ El sistema debe permitir registrar la realización de la reunión, eliminando los *horarios disponibles* de los convocados.
✓ El sistema debe mantener los *horarios disponibles* de los *convocados*.
✓ El sistema debe mostrar la existencia de fechas comunes disponibles entre los *horarios disponibles* de los *convocados* y del *convocante*.
✓ El sistema debe registras la fecha seleccionada por el *convocante* para la reunión.
✓ El sistema debe mostrar los espacios disponibles para la fecha seleccionada y la disponibilidad de *material físico* requerido.
✓ El sistema debe registrar el *lugar físico de la reunión*.
✓ El sistema debe permitir imprimir el *listado para convocatoria*.
✓ El sistema debe permitir la impresión de copias del *temario* según la cantidad de *convocados*.
✓ El sistema debe mantener la confirmación de concurrencia a la reunión, la no concurrencia o los datos del *reemplazante*.
✓ El sistema debe mantener el *material físico* reservado.
✓ El sistema debe registrar la cancelación de una *reunión*.
✓ El sistema debe permitir la actualización de datos de la *reunión*.

REQUISITOS NO FUNCIONALES
✓ El sistema debe construirse con una capa de datos que inicialmente maneje la BD Access pero que permita su fácil cambio a alguna otra BD (RNF extraído de una Ficha de Información Anticipada). ✓ El sistema debe calcular los *horarios disponibles* comunes en menos de 10 segundos el 90% de las veces.

LISTA DE REQUISITOS CON ALTA REINGENIERÍA DE LOS PROCESOS DEL NEGOCIO

REQUISITOS FUNCIONALES
✓ El sistema debe verificar nombre de usuario y clave para alta de *agendas*, alta de *convocados* externos, alta de espacios disponibles y alta de *material físico* disponible. ✓ El sistema debe mantener datos de empleados con *agenda*: nombre y apellido, e-mail, teléfono, cargo y sector. ✓ El sistema debe generar y mantener una *agenda de reuniones* por empleado. ✓ El sistema debe enviar un mensaje de error y reitera el pedido 3 veces al ingresar nombre de usuario y clave. ✓ El sistema debe verificar si el usuario ya tiene abierta una *agenda*. ✓ El sistema debe verificar los datos ingresados del empleado que no sean inválidos o nulos para la apertura de una *agenda*. ✓ El sistema debe mantener los datos de *convocados* externos: nombre y apellido, e-mail, teléfono, empresa, cargo y sector. ✓ El sistema debe verificar que los datos ingresados del *convocado* externo no sean inválidos o nulos o que ya existentes. ✓ El sistema debe mantener los datos de espacios disponibles: ubicación, capacidad de personas y si es anexo o no. ✓ El sistema debe verificar que los datos ingresados del espacio disponible no sean inválidos o nulos o ya existentes. ✓ El sistema debe mantener los datos del *material físico*: nombre y descripción. ✓ El sistema debe verificar que los datos ingresados del *material físico* no sean inválidos o nulos o ya existentes. ✓ El sistema debe armar y mostrar el *cronograma de reuniones* para un período solicitado, buscando las *reuniones* con *fecha establecida* en dicho período. ✓ El sistema debe mantener los datos de la *reunión*: el *objetivo*, los *temas*, los *convocados*, el *material a presentar*, el *material para repartir*, el *material físico*, si el *convocante* va a ser un *participante* de la *reunión* y si se requiere *confirmación de asistencia* a la *reunión*. ✓ El sistema debe mostrar los datos de una *reunión* a pedido.

- ✓ El sistema debe permitir consultar la *disponibilidad de espacio* para un período dado y dependiendo de la cantidad de *participantes*.
- ✓ El sistema debe permitir consultar la disponibilidad de *material físico* para un período dado.
- ✓ El sistema debe mantener por cada *reunión*: *fecha establecida*, *horario*, *lugar* y *temario* (opcional)
- ✓ El sistema debe permitir reservar *espacio físico* disponible para una *fecha establecida* y *horario*.
- ✓ El sistema debe permitir reservar *material físico* disponible para una *fecha establecida* y *horario*.
- ✓ El sistema debe permitir registrar la asistencia de cada *participante* a una *reunión*.
- ✓ El sistema debe permitir registrar la realización de la *reunión*, liberando el *espacio físico* y el *material físico* y eliminando los *horarios disponibles* de los *convocados* externos.
- ✓ El sistema debe permitir registrar un *reemplazante* en el momento de realización de la *reunión*.
- ✓ El sistema debe permitir registrar los *horarios disponibles* de los *convocados* externos para un período dado.
- ✓ El sistema debe determinar los *horarios disponibles* comunes a todos los *participantes* para un período dado.
- ✓ El sistema debe mostrar dichos *horarios disponibles* comunes.
- ✓ El sistema debe permitir consultar espacios disponibles y la disponibilidad de *material físico* requerido para la fecha seleccionada.
- ✓ El sistema debe permitir seleccionar el *lugar físico de la reunión*, en base a la *disponibilidad de espacio* y la disponibilidad de *material físico* requerido para la *fecha de la reunión* establecida.
- ✓ El sistema debe enviar a través de un medio de comunicación, a cada *convocado* la *fecha establecida*, *horario*, *lugar* de la *reunión*, el *temario* y el medio de comunicación de respuesta
- ✓ El sistema debe lanzar automáticamente la *convocatoria* mínimo 5 días antes de la *fecha establecida*.
- ✓ El sistema debe pedir la *confirmación de asistencia* a la *reunión* a cada *convocado* si se estipuló para la *reunión*.
- ✓ El sistema debe registrar el envío de la *convocatoria* en el *listado de convocados*.
- ✓ El sistema debe permitir que la *secretaria* registre la frecuencia en la que debe realizarse el *recordatorio* para la *reunión*.
- ✓ El sistema debe registrar la confirmación, ausencia o el reemplazo a una *reunión* por parte de un *convocado* externo.
- ✓ El sistema debe registrar en el *listado de convocados* la confirmación, ausencia o los datos del *reemplazante* para el caso de un *convocado* externo.
- ✓ El sistema debe verificar la posibilidad de reemplazar a un *convocado* externo.
- ✓ El sistema debe registrar la asistencia, ausencia o reemplazo a una *reunión* por parte de un *convocado* perteneciente a la organización.
- ✓ El sistema debe verificar que si el *convocado* informa su ausencia entonces éste no iba a exponer ningún *tema*.
- ✓ El sistema debe permitir la carga de datos de un *reemplazante* de un *convocado* si está permitida la asignación de un *reemplazante* y dicho *reemplazante* tiene abierta una *agenda de reuniones*.
- ✓ El sistema debe registrar la *reunión* en la *agenda* del *convocado* si éste asistirá a la *reunión*.

- ✓ El sistema debe registrar la *reunión* en la *agenda* del *reemplazante* si el *convocado* perteneciente a la organización asigna un *reemplazante*.
- ✓ **Si** el *convocado* no asiste o envía un *reemplazante* y ya había avisado su asistencia**,** el sistema debe registrar la baja de la *reunión* en la *agenda* del *convocado*.
- ✓ El sistema debe registrar en el *listado de convocados* la confirmación, ausencia o los datos del *reemplazante* para el caso de un convocado perteneciente a la organización.
- ✓ El sistema debe mantener registro de la reserva del *material físico*.
- ✓ El sistema debe verificar la existencia del *material físico* solicitado.
- ✓ El sistema debe verificar la disponibilidad del *material físico* solicitado para la fecha *establecida*.
- ✓ El sistema debe enviar por algún medio de comunicación, un *recordatorio* a cada *convocado* sobre la *fecha establecida*, *horario* y *lugar* de la *reunión*, utilizando el *listado para convocatoria* y según la frecuencia de *recordatorio*.
- ✓ El sistema debe verificar en el *listado de convocados* aquellos *convocados* que no avisaron sobre su asistencia y cuyo aviso es requerido.
- ✓ **Si** el *convocado* no ha efectuado el *aviso de concurrencia* y se requiere *confirmación de asistencia***,** el sistema debe solicitar por el mismo medio de comunicación la *confirmación de asistencia* a la *reunión*.
- ✓ El sistema debe registrar el *recordatorio* al *convocado* en el *listado para convocatoria*.
- ✓ El sistema debe permitir al *convocante* ingresar una nueva posible *fecha de la reunión*.
- ✓ El sistema debe permitir la consulta sobre la *disponibilidad de espacio* para una fecha dada.
- ✓ El sistema debe permitir la consulta sobre la disponibilidad del *material físico* para una fecha dada.
- ✓ El sistema debe registrar la cancelación de una *reunión*.
- ✓ **Si** se realizó la *convocatoria***,** el sistema debe enviar por algún medio de comunicación el *aviso de cancelación* a cada *convocado* o *reemplazante*, utilizando el *listado para convocatoria*.
- ✓ **Si** se realizó la *convocatoria*, el sistema debe registrar el *aviso de cancelación* en el *listado para convocatoria*.
- ✓ El sistema debe permitir anular la reserva de *espacio físico*.
- ✓ El sistema debe permitir anular la reserva de *material físico*.
- ✓ El sistema debe permitir actualizar los datos de la *reunión* por un traslado de fecha: nueva fecha, nuevo *horario*, nuevo *lugar*.
- ✓ **Si** se realizó la *convocatoria***,** el sistema debe avisar por algún medio de comunicación el cambio de la *fecha de la reunión*, *horario* y/o *lugar* a cada *convocado*, utilizando el *listado para convocatoria*.
- ✓ **Si** se realizó la *convocatoria***,** el sistema debe registrar la comunicación de t*raslado de fecha* en el *listado para convocatoria*.
- ✓ **Si** hay cambio del *lugar de la reunión***,** el sistema debe anular la reserva de *espacio físico* y registrar una nueva reserva de *lugar*.
- ✓ El sistema debe permitir anular la reserva de *material físico* para la fecha anterior y reservarlo para la nueva *fecha de la reunión*.
- ✓ El sistema debe permitir actualizar los datos de la *reunión*: *lugar de la reunión*, *temario*, *material a presentar*, *material para repartir* y/o *material físico*.
- ✓ **Si** se realizó la *convocatoria***,** el sistema debe avisar por algún medio de comunicación los *cambios en los requerimientos de la reunión* a cada *convocado*,

utilizando el *listado para convocatoria*.

- ✓ **Si** se realizó la *convocatoria***,** el sistema debe registrar la comunicación de cambio en el *listado para convocatoria*.
- ✓ **Si** hay cambio del *lugar de la reunión***,** el sistema debe anular la reserva de *espacio físico* y realizar una nueva reserva de *lugar*.
- ✓ **Si** hay cambio del *lugar de la reunión***,** el sistema debe anular la reserva de *material físico* para la fecha anterior y reservarlo para la nueva *fecha de la reunión*.
- ✓ **Si** hay cambio del *material físico***,** el sistema debe anular la reserva de *material físico* y reservar el nuevo *material físico*.
- ✓ **Si** hay cambio de los *temas* a tratar**,** el sistema debe enviar por algún medio de comunicación el nuevo *temario* a cada *convocado*.
- ✓ El sistema debe permitir la consulta de espacios disponibles en el edificio anexo para la *fecha establecida* y el *horario*.
- ✓ El sistema debe permitir registrar en el sistema el *lugar físico de la reunión* proveniente del edificio anexo.

REQUISITOS NO FUNCIONALES

- ✓ El sistema debe trabajar bajo una red que soporte un máximo de 100 usuarios trabajando simultáneamente sobre el sistema (RNF extraído de una Ficha de Información Anticipada).
- ✓ El sistema debe poder œnectarse por algún medio de comunicación (e-mail) con todos los *convocados* a una *reunión* (RNF extraído del uso del recurso *medio de comunicación* en varios escenarios).
- ✓ El sistema debe construirse con una capa de datos que inicialmente maneje la BD Access pero que permita su fácil cambio a alguna otra BD (RNF extraído de una Ficha de Información Anticipada).
- ✓ Todo usuario del sistema debe ser un usuario de la red de la organización (RNF extraído de una Ficha de Información Anticipada).
- ✓ El personal tiene asignado un nombre de usuario y clave que el sistema debe reconocer.
- ✓ El empleado debe estar autorizado a tener una *agenda de reuniones*.
- ✓ El sistema envía mensajes informativos ante fallas en la carga de datos por los usuarios.
- ✓ El sistema debe mostrar el *cronograma de reuniones* en menos de 30 segundos el 100% de las veces.
- ✓ El sistema debe calcular los *horarios disponibles* comunes en menos de 10 segundos el 90% de las veces.

Apéndice E
Modelo de Trazas

«The context in which software is developed is a fast changing arena where new technologies appear and disappear and stakeholders change their expectations.»

J. C.S. Leite, J.H. Doorn, "Perspectives on Software Requirements", Kluwer Academic Publishers, 2004

E.1. Introducción

Se presenta en este apéndice una versión preliminar de un tema estudiado: trazas para versionado de los modelos de SDRES, basado en un modelo simple de trazas que provee una gran facilidad de registro de las mismas y ductibilidad para la obtención de información histórica.

El modelo de datos de la Figura E-1 representa trazas históricas dentro de CORB basada en el apilamiento de trazas, donde no hay versiones intermedias de trabajo, sino que se genera una nueva versión al cierre de un modelo o documento. Se dispone de la versión anterior, la versión actual correspondiente al cierre y un archivo de trazas que describe todos los cambios realizados desde la versión inicial hasta la nueva.

El manejo de trazas por apilamiento permite fácilmente retroceder a versiones anteriores de un modelo observando los cambios realizados sobre las entidades de cada modelo. Un modelo básico de trazas se puede manejar con información tal como:

- Fecha
- Responsable
- Causa
- Número de Pedido de Cambio
- Lista de símbolos del LEL afectados
- Versión del LEL afectado
- Lista de EA del Conjunto de EA afectados
- Versión del Conjunto de EA afectado
- Lista de EF del Conjunto de EF afectados
- Versión del Conjunto de EF afectado
- Lista de requisitos del SRS afectados
- Versión del SRS afectado

En esta tesis, se propone adicionar un meta-modelo que facilite el registro y extracción de trazas, mediante un conjunto de operaciones que reflejen todos los posibles cambios en los modelos de SDRES. Luego, el nivel de detalle en la historia de cambios dependerá del meta-modelo que se construya durante la definición de trazas.

En el párrafo anterior se está haciendo mención a las tres actividades del modelado de trazas [Pinheiro 04]: definición, producción y extracción de trazas. La primera consiste en la especificación de las trazas y los ítems a versionar. La producción consiste en la captura o registro de trazas, tarea que debe realizarse en oportunidad de la ocurrencia del cambio, y en la forma especificada previamente. La extracción de trazas consiste en la obtención de información proveniente de las trazas que permita seguir el camino de cambios hasta la versión anterior de un ítem.

El meta-modelo está compuesto de dos entidades <<Tipo de Operación>> y <<Entidad>> (ver Figura E-1), las cuales permiten determinar qué vínculos dentro del modelo o entre modelos del CORB se pueden especificar para luego registrar y extraer información de los cambios. Lo establecido en la definición de trazas debe estar aprobado previo al uso del sistema de trazas, pues cambios en este meta-modelo, una vez en operación el sistema de trazas, pueden producir pérdida o inconsistencia en la información.

<<Entidad>> representa todo ítem que puede sufrir cambios y por ende ser versionado, e incluye modelos, componentes y sub-componentes de modelos. <<Tipo de Operación>> representa un tipo de cambio que puede sufrir una <<Entidad>> correspondiente a la versión de un dado modelo. Si la entidad es parte de otra entidad, entonces se debe identificar la entidad principal o clave. Además un cambio puede afectar dos entidades del mismo modelo o entidades de distinto modelo, es decir, se tratan de operaciones relacionadas con vínculos semánticos inter o extra modelo (dentro del CORB). Con este meta-modelo se facilitará posteriormente la carga de trazas, guiando en cada operación el tipo de información a ingresar.

Cada cambio en los modelos de requisitos debe producir un evento en el sistema de trazas que genere una operación que refleje el cambio ocurrido en los modelos. El registro de trazas es una tarea fundamental que debe realizarse en tiempo y forma, sino se inhabilita todo control de cambios y, es por ello que el meta-modelo procura facilitar el registro de las trazas. La captura de trazas se hace a través de transacciones que pueden involucrar una única operación o un conjunto de operaciones relacionados con una solicitud de cambio, e incluso una solicitud puede implementarse en varias transacciones. Ésta es una política que debe establecerse previamente para luego realizar consultas de trazas bajo ciertas premisas.

Las trazas se apilan en el orden de almacenamiento, lo cual indica que al desapilar, se retrocede en los pedidos de cambios. No obstante ello, es posible contar con vistas y consultas que permitan obtener diversa información al extraer trazas. En la Figura E-2 se muestra una especificación de vista y algunos posibles filtros para extraer trazas.

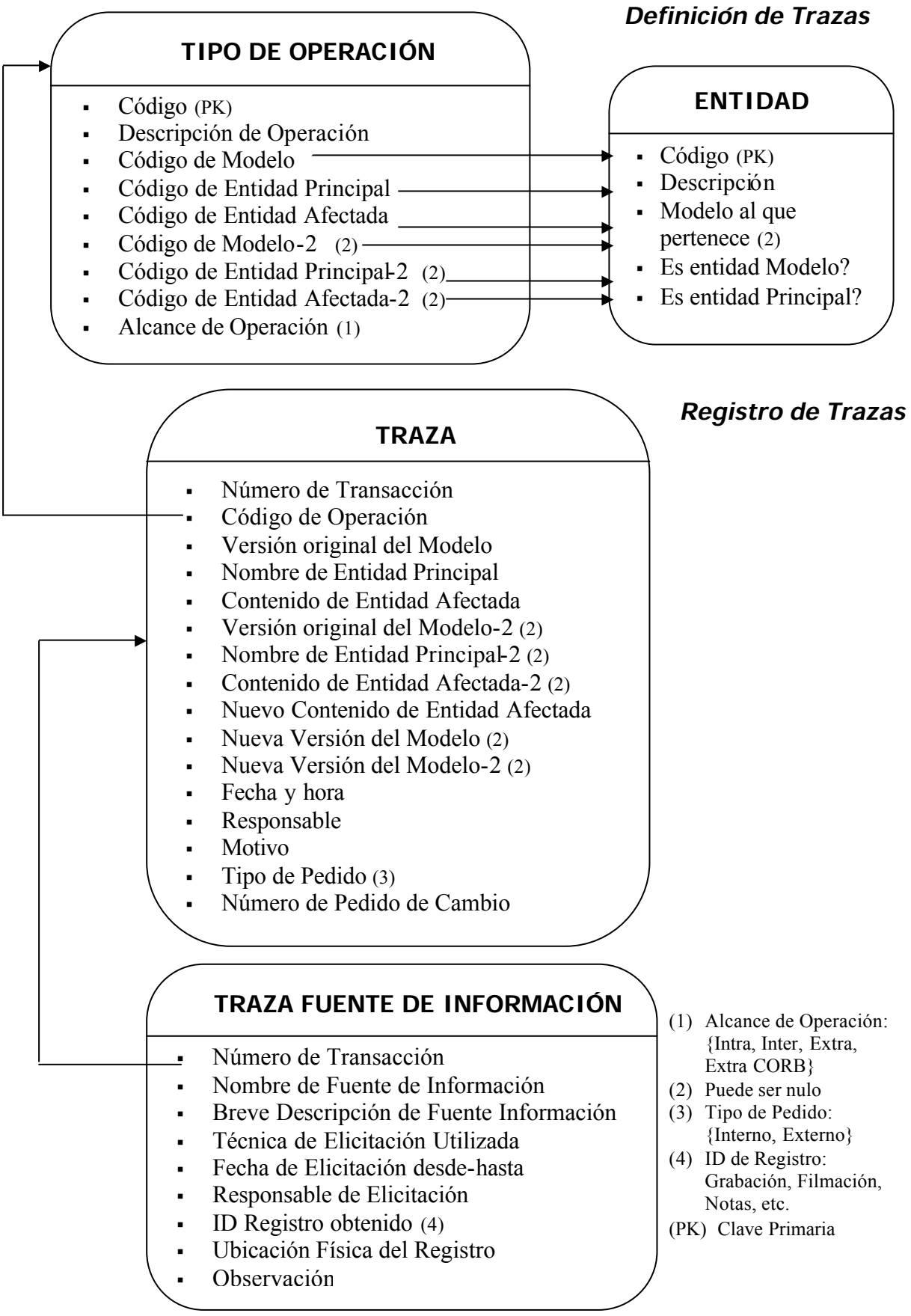


Figura E-1. Modelo de Trazas para Versionado

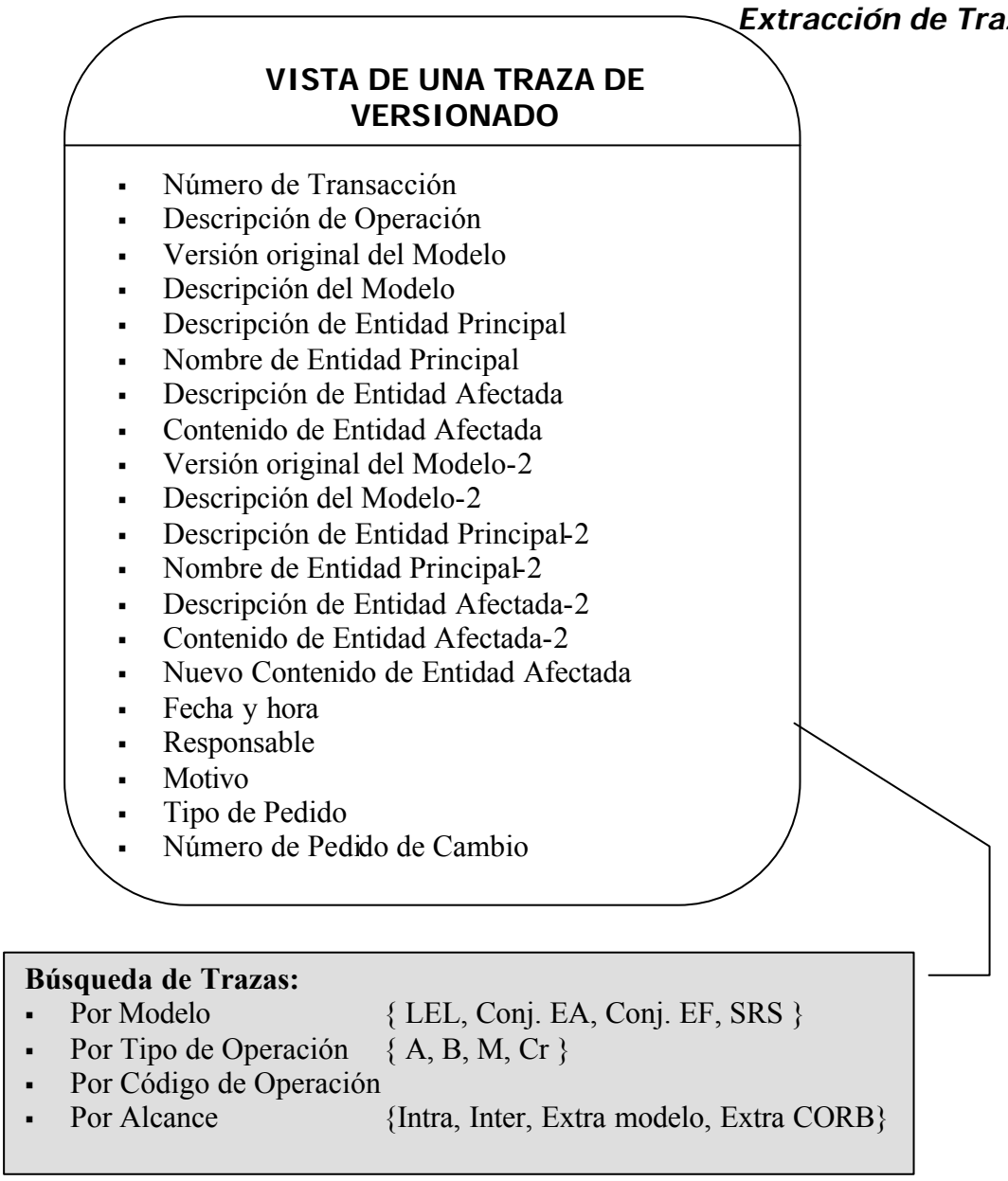


Figura E-2. Vista de una Traza

El Sistema de Trazas permite realizar las siguientes actividades:

- ⇨ Definición del Meta-Modelo
- ⇨ Carga de Trazas
- ⇨ Extracción de Trazas
- ⇨ Cierre de Versión por modelo

La Definición del Meta-Modelo incluye las sub-actividades:

i) Carga de los tipos de ítems de configuración, es decir, los modelos de requisitos y los componentes cuyos cambios se desean controlar. Se trata de <<Entidad>>.

ii) Carga de los tipos de operaciones a realizar sobre los ítems de

configuración. Básicamente son operaciones de creación de modelos; alta, baja y modificación de componentes; y alta y baja de vínculos de dependencia entre componentes del modelo y entre modelos. Las operaciones pueden definirse a nivel elemental de cada entidad del modelo y cada parte de la entidad, o a nivel de modelos y de entidades más importantes. Se trata de <<Operación>>.

En las secciones siguientes del anexo, se presentan dos ejemplos de este Meta-Modelo, donde un ejemplo presenta una gran segmentación de operaciones, e inversamente el otro ejemplo muestra unas pocas operaciones, siendo ambos ejemplos aplicables a los modelos de SDRES.

La Carga de Trazas consiste en registrar las operaciones que se realizan sobre los modelos de requisitos con motivo de los cambios solicitados, fundamentando cada cambio. En general un pedido de cambio, involucrará varias operaciones y probablemente abarcará más de un modelo.

La Extracción de Trazas consiste en la captura de las trazas registradas para recorrer la historia de los cambios para un modelo dado desde una versión dada a otra, o para un componente o sub-componente dado del modelo o para controlar la resolución de un pedido de cambio u otras consultas, que pueden estar predefinidas a través de vistas o consultas, o que pueden resolverse interactivamente. Pueden incluirse entonces sub-actividades de apoyo para la generación de vistas y consultas de aquellas más relevantes o frecuentes.

El Cierre de Versión permite la generación de una nueva versión de un modelo de requisitos, la historia de los cambios previos se mantienen adosándoles el nuevo número de versión, y a partir de esta decisión todo nuevo pedido de cambio provoca el registro de trazas pero se liberará recién a partir del próximo cierre.

E.2. Ejemplos

Se presenta en este apéndice dos ejemplos de cómo establecer trazas bien diferentes a partir del meta-modelo propuesto. En el primer ejemplo, el contenido de la tabla Tipo de Operación incluye un alto grado de disgregación de operaciones, en la gran mayoría se puede decir que son operaciones primitivas, aunque por ejemplo, las operaciones con manejo de vínculos semánticos podrían aún descomponerse para manejar el vínculo a nivel de sub-componente en el que puede presentarse. En el primer ejemplo se puede mencionar la operación: el alta de una referencia a un dado símbolo en un dado EA, pero podría además refinarse esta operación para indicar el alta de una referencia a un dado símbolo en un episodio dado de un EA dado.

Pero también es viable un conjunto de operaciones más compacto que agilice su registro. Esta elección dependerá del tipo de aplicación con el control requerido sobre ella, los recursos disponibles para el proyecto, y las

necesidades del grupo de desarrollo acorde con lo pactado con el cliente.

En el segundo ejemplo, se presenta un conjunto mínimo de operaciones con el cual también puede rastrearse la historia de los cambios con menos detalles pero con más agilidad en el registro. Para este conjunto menor de operaciones, se requiere también un conjunto menor de elementos o se puede mantener siempre el conjunto completo de elementos.

Cabe notar que las operaciones de reorganización de SDRES: Factorizar, Aplanar, Dividir, Consolidar, Separar y Fusionar, involucran una o más operaciones de Alta de un EA y una o más operaciones de Baja de un EA o Modificación de EA, junto con Altas y/o Bajas de vínculos a sub-escenarios. Es decir, estas seis operaciones de reorganización se pueden descomponer en operaciones menores del meta modelo, junto también con la integración de escenarios. Estas operaciones podrían servir de guía para facilitar la realización de la actividad Organizar pero no son necesarias en sí mismas como operaciones primitivas a registrar para el versionado. Lo mismo ocurre para el conjunto de EF. Como una transacción involucra un conjunto de operaciones, fácilmente una transacción puede ser una operación de reorganización, la cual a su vez de descompone en operaciones del meta-modelo. De esta forma, es fácilmente rastreable toda operación de reorganización, lo cual es válido también para la integración de escenarios.

---------- en blanco ----------

E.3. Codificación de Entidades – EJEMPLO 1

Código	Descripción	Modelo al que pertenece	Es entidad Modelo ?	Es entidad Principal ?
1.	LEL	-	Sí	No
2.	Conjunto de EA	-	Sí	No
3.	Conjunto de EF	-	Sí	No
4.	SRS	-	Sí	No
5.	Símbolo	1	No	Sí
6.	Nombre de Símbolo	1	No	No
7.	Noción	1	No	No
8.	Impacto	1	No	No
9.	EA	2	No	Sí
10.	Título	2	No	No
11.	Objetivo	2	No	No
12.	Ubicación Geográfica	2	No	No
13.	Ubicación Temporal	2	No	No
14.	Precondición	2	No	No
15.	Actor	2	No	No
16.	Recurso	2	No	No
17.	Episodio	2	No	No
18.	Restricción	2	No	No
19.	Excepción	2	No	No
20.	Causa de Excepción	2	No	No
21.	Lista de Episodios Excepción	2	No	No
22.	Solución a Excepción	2	No	No
23.	EF	3	No	Sí
24.	Título	3	No	No
25.	Objetivo	3	No	No
26.	Ubicación Geográfica	3	No	No
27.	Ubicación Temporal	3	No	No
28.	Precondición	3	No	No
29.	Actor	3	No	No
30.	Recurso	3	No	No
31.	Episodio	3	No	No
32.	Restricción	3	No	No
33.	Excepción	3	No	No
34.	Causa de Excepción	3	No	No
35.	Lista de Episodios Excepción	3	No	No
36.	Solución a Excepción	3	No	No
37.	Requisito	4	No	Sí
38.	Nombre	4	No	No
39.	Descripción	4	No	No
40.	Tipo de Requisito	4	No	No
41.	Prioridad	4	No	No

Código	Descripción	Modelo al que pertenece	Es entidad Modelo ?	Es entidad Principal ?
42.	Criticidad	4	No	No
43.	Estado	4	No	No
44.	Riesgo	4	No	No
45.	Origen	4	No	No
46.	Fundamento	4	No	No

E.4. Tipos de Operaciones – EJEMPLO 1

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
1	Creación del LEL	Intra	LEL	LEL	LEL			
2	Alta de símbolo	Intra	LEL	Símbolo	Símbolo			
3	Baja de símbolo	Intra	LEL	Símbolo	Símbolo			
4	Alta de nombre de símbolo	Intra	LEL	Símbolo	Nombre			
5	Baja de nombre de símbolo	Intra	LEL	Símbolo	Nombre			
6	Modificación de nombre de símbolo	Intra	LEL	Símbolo	Nombre			
7	Alta de noción de símbolo	Intra	LEL	Símbolo	Noción			
8	Baja de noción de símbolo	Intra	LEL	Símbolo	Noción			
9	Modificación de noción de símbolo	Intra	LEL	Símbolo	Noción			
10	Alta de impacto de símbolo	Intra	LEL	Símbolo	Impacto			
11	Baja de impacto de símbolo	Intra	LEL	Símbolo	Impacto			
12	Modificación de impacto de símbolo	Intra	LEL	Símbolo	Impacto			
300	Alta de link a símbolo	Inter.	LEL	Símbolo	Símbolo		Símbolo	Símbolo
301	Baja de link a símbolo	Inter.	LEL	Símbolo	Símbolo		Símbolo	Símbolo
302	Alta de link de símbolo a modelos externos al CORB	Extra CORB	LEL	Símbolo	Símbolo	XX	XX	XX
303	Baja de link de símbolo a modelos externos al CORB	Extra CORB	LEL	Símbolo	Símbolo	XX	XX	XX
13	Creación del Conjunto de EA	Intra	Conj.EA	Conj.EA	Conj.EA			
14	Alta de un EA	Intra	Conj.EA	EA	EA			
15	Baja de un EA	Intra	Conj.EA	EA	EA			
16	Modificación de Título	Intra	Conj.EA	EA	Título			

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
17	Modificación de Objetivo	Intra	Conj.EA	EA	Objetivo			
18	Alta de Ubicación Geográfica	Intra	Conj.EA	EA	Ubicac.Geog.			
19	Baja de Ubicación Geográfica	Intra	Conj.EA	EA	Ubicac.Geog.			
20	Modificación de Ubicación Geográfica	Intra	Conj.EA	EA	Ubicac.Geog.			
21	Alta de Ubicación Temporal	Intra	Conj.EA	EA	Ubicac.Temp.			
22	Baja de Ubicación Temporal	Intra	Conj.EA	EA	Ubicac.Temp.			
23	Modificación de Ubicación Temporal	Intra	Conj.EA	EA	Ubicac.Temp.			
24	Alta de Precondición	Intra	Conj.EA	EA	Precondición			
25	Baja de Precondición	Intra	Conj.EA	EA	Precondición			
26	Modificación de Precondición	Intra	Conj.EA	EA	Precondición			
27	Alta de Actor	Intra	Conj.EA	EA	Actor			
28	Baja de Actor	Intra	Conj.EA	EA	Actor			
29	Modificación de Actor	Intra	Conj.EA	EA	Actor			
30	Alta de Recurso	Intra	Conj.EA	EA	Recurso			
31	Baja de Recurso	Intra	Conj.EA	EA	Recurso			
32	Modificación de Recurso	Intra	Conj.EA	EA	Recurso			
33	Alta de un Episodio	Intra	Conj.EA	EA	Episodio			
34	Baja de un Episodio	Intra	Conj.EA	EA	Episodio			
35	Modificación de un Episodio	Intra	Conj.EA	EA	Episodio			
36	Alta de Restricción a un Episodio	Intra	Conj.EA	EA	Restricción			
37	Baja de Restricción a un Episodio	Intra	Conj.EA	EA	Restricción			
38	Modificación de Restricción a un Episodio	Intra	Conj.EA	EA	Restricción			
39	Alta de Excepción	Intra	Conj.EA	EA	Excepción			

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
40	Baja de Excepción	Intra	Conj.EA	EA	Excepción			
41	Modificación de Causa de Excepción	Intra	Conj.EA	EA	Causa de Excepción			
42	Modificación de Lista de Episodios de Excepción	Intra	Conj.EA	EA	Lista Episodio Excepción			
43	Modificación de Solución de Excepción	Intra	Conj.EA	EA	Solución de Excepción			
304	Alta de link a sub-escenario en EA	Inter	Conj.EA	EA	Episodio		EA	EA
305	Baja de link a sub-escenario en EA	Inter	Conj.EA	EA	Episodio		EA	EA
306	Alta de link a escenario excepción en EA	Inter	Conj.EA	EA	Excepción		EA	EA
307	Baja de link a escenario excepción en EA	Inter	Conj.EA	EA	Excepción		EA	EA
308	Alta de link a símbolo en EA	Extra	Conj.EA	EA	EA	LEL	Símbolo	Símbolo
309	Baja de link a símbolo en EA	Extra	Conj.EA	EA	EA	LEL	Símbolo	Símbolo
310	Modificación de nombre de símbolo en EA	Extra	Conj.EA	EA	EA	LEL	Símbolo	Símbolo
311	Alta de link bidireccional a EF por derivación de EA	Extra	Conj.EA	EA	EA	Conj. EF	EF	EF
312	Baja de link bidireccional a EF por derivación de EA	Extra	Conj.EA	EA	EA	Conj. EF	EF	EF
313	Alta de link de EA a modelos externos al CORB	Extra CORB	Conj.EA	EA	EA	XX	XX	XX
314	Baja de link de EA a modelos externos al CORB	Extra CORB	Conj.EA	EA	EA	XX	XX	XX
44	Creación del Conjunto de EF	Intra	Conj. EF	Conj. EF	Conj. EF			
45	Alta de un EF	Intra	Conj.EF	EF	EF			

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
46	Baja de un EF	Intra	Conj.EF	EF	EF			
47	Modificación de Título	Intra	Conj.EF	EF	Título			
48	Modificación de Objetivo	Intra	Conj.EF	EF	Objetivo			
49	Alta de Ubicación Geográfica	Intra	Conj.EF	EF	Ubicac.Geog.			
50	Baja de Ubicación Geográfica	Intra	Conj.EF	EF	Ubicac.Geog.			
51	Modificación de Ubicación Geográfica	Intra	Conj.EF	EF	Ubicac.Geog.			
52	Alta de Ubicación Temporal	Intra	Conj.EF	EF	Ubicac.Temp.			
53	Baja de Ubicación Temporal	Intra	Conj.EF	EF	Ubicac.Temp.			
54	Modificación de Ubicación Temporal	Intra	Conj.EF	EF	Ubicac.Temp.			
55	Alta de Precondición	Intra	Conj.EF	EF	Precondición			
56	Baja de Precondición	Intra	Conj.EF	EF	Precondición			
57	Modificación de Precondición	Intra	Conj.EF	EF	Precondición			
58	Alta de Actor	Intra	Conj.EF	EF	Actor			
59	Baja de Actor	Intra	Conj.EF	EF	Actor			
60	Modificación de Actor	Intra	Conj.EF	EF	Actor			
61	Alta de Recurso	Intra	Conj.EF	EF	Recurso			
62	Baja de Recurso	Intra	Conj.EF	EF	Recurso			
63	Modificación de Recurso	Intra	Conj.EF	EF	Recurso			
64	Alta de un Episodio	Intra	Conj.EF	EF	Episodio			
65	Baja de un Episodio	Intra	Conj.EF	EF	Episodio			
66	Modificación de un Episodio	Intra	Conj.EF	EF	Episodio			
67	Alta de Restricción a un Episodio	Intra	Conj.EF	EF	Restricción			
68	Baja de Restricción a un Episodio	Intra	Conj.EF	EF	Restricción			
69	Modificación de Restricción a un	Intra	Conj.EF	EF	Restricción			

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
	Episodio							
70	Alta de Excepción	Intra	Conj.EF	EF	Excepción			
71	Baja de Excepción	Intra	Conj.EF	EF	Excepción			
72	Modificación de Causa de Excepción	Intra	Conj.EF	EF	Causa de Excepción			
73	Modificación de Lista de Episodios Excepción	Intra	Conj.EF	EF	Lista Episodio Excepción			
74	Modificación de Solución de Excepción	Intra	Conj.EF	EF	Solución de Excepción			
315	Alta de link a sub-escenario en EF	Inter	Conj.EF	EF	Episodio		EF	EF
316	Baja de link a sub-escenario en EF	Inter	Conj.EF	EF	Episodio		EF	EF
317	Alta de link a escenario excepción en EF	Inter	Conj.EF	EF	Excepción		EF	EF
318	Baja de link a escenario excepción en EF	Inter	Conj.EF	EF	Excepción		EF	EF
319	Alta de link a símbolo en EF	Extra	Conj.EF	EF	EF	LEL	Símbolo	Símbolo
320	Baja de link a símbolo en EF	Extra	Conj.EF	EF	EF	LEL	Símbolo	Símbolo
321	Modificación de nombre de símbolo en EF	Extra	Conj.EF	EF	EF	LEL	Símbolo	Símbolo
322	Alta de link bidireccional a SRS por explicitar requisito de EF	Extra	Conj.EF	EF	EF	SRS	Requisito	Requisito
323	Baja de link bidireccional a SRS por explicitar requisito de EF	Extra	Conj. EF	EF	EF	SRS	Requisito	Requisito
324	Alta de link de EF a modelos externos al CORB	Extra CORB	Conj. EF	EF	EF	XX	XX	XX
325	Baja de link de EF a modelos externos al CORB	Extra CORB	Conj. EF	EF	EF	XX	XX	XX

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
75	Creación del SRS	Intra	SRS	SRS	SRS			
76	Alta de Requisito	Intra	SRS	Requisito	Requisito			
77	Baja de Requisito	Intra	SRS	Requisito	Requisito			
78	Modificación de Nombre	Intra	SRS	Requisito	Nombre			
79	Modificación de Descripción	Intra	SRS	Requisito	Descripción			
80	Modificación de Tipo de Requisito	Intra	SRS	Requisito	Tipo Requisito			
81	Modificación de Prioridad	Intra	SRS	Requisito	Prioridad			
82	Modificación de Criticidad	Intra	SRS	Requisito	Criticidad			
83	Modificación de Estado	Intra	SRS	Requisito	Estado			
84	Modificación de Riesgo	Intra	SRS	Requisito	Riesgo			
85	Modificación de Origen	Intra	SRS	Requisito	Origen			
86	Modificación de Fundamento	Intra	SRS	Requisito	Fundamento			
326	Alta de link a Requisito	Inter	SRS	Requisito	Requisito			
327	Baja de link a Requisito	Inter	SRS	Requisito	Requisito			
328	Alta de link a símbolo en Requisito	Extra	SRS	Requisito	Requisito	LEL	Símbolo	Símbolo
329	Baja de link a símbolo en Requisito	Extra	SRS	Requisito	Requisito	LEL	Símbolo	Símbolo
330	Modificación de nombre de símbolo en Requisito	Extra	SRS	Requisito	Requisito	LEL	Símbolo	Símbolo
331	Alta de link de Requisito a modelos externos al CORB	Extra CORB	SRS	Requisito	Requisito	XX	XX	XX
332	Baja de link de Requisito a modelos externos al CORB	Extra CORB	SRS	Requisito	Requisito	XX	XX	XX

XX: Entidad que depende del modelo extra CORB.

E.5. Codificación de Entidades – EJEMPLO 2

Código	Descripción	Modelo al que pertenece	Es entidad Modelo ?	Es entidad Principal ?
1.	LEL	-	Sí	No
2.	Conjunto de EA	-	Sí	No
3.	Conjunto de EF	-	Sí	No
4.	SRS	-	Sí	No
5.	Símbolo	1	No	Sí
6.	EA	2	No	Sí
7.	EF	3	No	Sí
8.	Requisito	4	No	Sí

E.6. Tipos de Operaciones – EJEMPLO 2

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
1	Creación del LEL	Intra	LEL	LEL	LEL			
2	Alta de símbolo	Intra	LEL	Símbolo	Símbolo			
3	Baja de símbolo	Intra	LEL	Símbolo	Símbolo			
4	Modificación de símbolo	Intra	LEL	Símbolo	Símbolo			
100	Alta de link de símbolo a modelos externos al CORB	Extra CORB	LEL	Símbolo	Símbolo	XX	XX	XX
101	Baja de link de símbolo a modelos externos al CORB	Extra CORB	LEL	Símbolo	Símbolo	XX	XX	XX
5	Creación del Conjunto de EA	Intra	Conj.EA	Conj. EA	Conj.EA			
6	Alta de un EA	Intra	Conj.EA	EA	EA			
7	Baja de un EA	Intra	Conj.EA	EA	EA			
8	Modificación de un EA	Intra	Conj.EA	EA	EA			
102	Alta de link de EA a modelos externos al CORB	Extra CORB	Conj.EA	EA	EA	XX	XX	XX
103	Baja de link de EA a modelos externos al CORB	Extra CORB	Conj.EA	EA	EA	XX	XX	XX
9	Creación del Conjunto de EF	Intra	Conj.EF	Conj.EF	Conj.EF			
10	Alta de un EF	Intra	Conj.EF	EF	EF			
11	Baja de un EF	Intra	Conj.EF	EF	EF			
12	Modificación de un EF	Intra	Conj.EF	EF	EF			
104	Alta de link de EF a modelos externos al CORB	Extra CORB	Conj.EF	EF	EF	XX	XX	XX
105	Baja de link de EF a modelos externos al CORB	Extra CORB	Conj.EF	EF	EF	XX	XX	XX

Cód	Descripción de Operación	Alcance	Modelo	Entidad Principal	Entidad Afectada	Modelo 2	Entidad Principal 2	Entidad Afectada 2
13	Creación del SRS	Intra	SRS	SRS	SRS			
14	Alta de Requisito	Intra	SRS	Requisito	Requisito			
15	Baja de Requisito	Intra	SRS	Requisito	Requisito			
16	Modificación de Requisito	Intra	SRS	Requisito	Requisito			
106	Alta de link de Requisito a modelos externos al CORB	Extra CORB	SRS	Requisito	Requisito	XX	XX	XX
107	Baja de link de Requisito a modelos externos al CORB	Extra CORB	SRS	Requisito	Requisito	XX	XX	XX

En este ejemplo, se debe aclarar que no hay eliminación de operaciones sino agrupamiento de operaciones, en operaciones más generales. Por ello, cabe notar que las operaciones de Modificación de una entidad incluyen alta, baja o modificación de vínculos dentro del CORB.

Apéndice F
Aplicación de Inspecciones en Escenarios

«… dos supersticiones muy españolas: la creencia de que el escritor debe usar todas las palabras del diccionario, la creencia de que en cada palabra el significado es lo esencial y nada importan su connotación y su ambiente.»

J. L. Borges, "Leopoldo Lugones", Emecé, 1997

F.1. Introducción

Se muestra a continuación los formularios y las guías de instrucción para completar cada formulario, de manera de realizar tanto la inspección intra escenarios como la inspección inter escenarios durante la etapa de Preparación por parte del inspector asignado. Luego se presentan los formularios que se completan durante la actividad Reunión para evaluar el material inspeccionado y el proceso mismo de inspección.

En el capítulo 6 se muestran algunos de estos formularios con datos extraídos de las inspecciones realizadas al "Sistema de Agenda de Reuniones". En [Leite 05] se presentan datos de inspección de varios casos de estudio que muestran las mejoras obtenidas en la calidad de los escenarios, a pesar del esfuerzo de inspección que es alrededor de un 25% del esfuerzo de construcción de los escenarios.

F.2. Intra Inspección

F.2.1. Guías de Instrucción Intra Inspección

Tabla 1. **REPORTE DE CONSISTENCIA INTERNA DE ESCENARIOS.**
***Objetivo**:* Detectar discrepancias, errores y omisiones en cada escenario de un punto de vista.
***Precondiciones**:* • Se debe disponer de un LEL clasificado. • El responsable de la verificación debe conocer el LEL. • Se debe disponer de una versión de los escenarios en un medio en el que se pueda agregar información.

***Proceso General**:*

- Numerar correlativamente los escenarios.
- Completar los formularios según el siguiente orden:
  - ♦ Formulario I
  - ♦ Para cada uno de los escenarios:
    - ◊ Formulario II
    - ◊ Formulario III
    - ◊ Formulario IV
    - ◊ Formulario V
    - ◊ Formulario VI
    - ◊ Formulario VII
    - ◊ Formulario VIII
    - ◊ Formulario IX

    Agregar en la numeración de cada formulario el número de escenario. El Formulario X se debe llenar al mismo tiempo que se llenan los anteriores, ya que toda observación, corrección o información dudosa debe registrarse en esta planilla, tan pronto como se detecte y debe hacerse referencia al formulario donde se originó.

- En el caso de los Formularios IV a VIII, y X, utilizar tantas hojas como sea necesario numerándolas correlativamente.
- Una vez que se han procesado todos los formularios de todos los escenarios, archivar los mismos siguiendo estrictamente los números de formulario.
- Una vez finalizada la verificación, revisar los comentarios registrados en los Formularios X y realizar en el Formulario XI un comentario general acerca de la calidad del conjunto de escenarios y sobre todo otro aspecto que corresponda.

***Análisis**:*

En todos los formularios, especialmente del III en adelante, es posible detectar inconvenientes de diferente naturaleza en los escenarios bajo estudio. En algunos casos, estos inconvenientes son subsanables en el sentido que con la información disponible es posible discernir con seguridad la corrección a realizar; mientras que en otros es necesario recurrir al autor de los mismos o eventualmente recabar información del Universo de Discurso.

Todo problema debe ser catalogado como subsanable sólo si existe una razonable confianza en la corrección que se propone, en caso contrario se lo debe considerar no subsanable.

Forma I. **CARÁTULA**.

***Objetivo**:* Identificar el proyecto y el conjunto de escenarios a consistir.

***Pasos**:*

- Registrar la información básica y disponible del proyecto y del conjunto de escenarios.
- Al finalizar el proceso, registrar el esfuerzo de verificación del conjunto de escenarios.

***Análisis**:*

No corresponde.

Forma II. **RESUMEN CUANTITATIVO DE UN ESCENARIO**.
***Objetivo**:* Brindar una idea general del volumen de información presente en un escenario.
***Pasos**:* • Registrar el nombre del escenario a inspeccionar. • Registrar en la columna CANTIDAD los totales para cada rubro. • Registrar en la columna RELACIÓN si lo indicado para cada rubro es o no coherente con el título del escenario. En el caso en que el objetivo o alguna parte del contexto no se relacione con el título evaluar la naturaleza semántica de la discrepancia y registrar la misma en el Formulario X. • Cuando la cantidad de actores sea notoriamente superior a la cantidad de recursos o viceversa, considerar la posibilidad que existan omisiones de actores o de recursos y registrar la misma en el Formulario X. • Numerar correlativamente los episodios (si es que no lo están). • Cuando la cantidad de episodios sea muy grande, considerar la posibilidad de encontrarse en presencia de un escenario demasiado abarcativo y registrarla en el Formulario X. • Cuando el cociente entre las referencias totales al LEL y las referencias únicas sea demasiado grande, considerar la posibilidad de encontrarse en presencia de un escenario innecesariamente detallado y registrar esta información en el Formulario X.
***Análisis**:* Los componentes del escenario deben ser coherentes entre sí y estar balanceados, de tal manera que todo el escenario debe quedar cubierto por su título y los restantes componentes del mismo deben contribuir al cumplimiento del objetivo enunciado. Por otra parte la cantidad de actores, recursos y episodios no debe mostrar una asimetría notable ya que esto indica algún tipo de vicio de construcción del escenario.

Forma III. **COMPROBACIÓN SINTÁCTICA DE UN ESCENARIO.**
***Objetivo**:* Verificar si los componentes de un escenario se encuentran correctamente escritos.
***Pasos**:* • Para cada uno de los componentes del escenario, comparar su texto con la sintaxis definida para el mismo. • Registrar en la columna FALTANTE toda omisión de elementos sintácticos obligatorios. • Registrar en la columna SOBRANTE todo elemento incluido en el componente que no se aparea con la sintaxis. • Para cada componente verificar si los faltantes registrados son irreparables a partir de los restantes componentes del escenario; verificar también si los sobrantes pueden ser eliminados con razonable seguridad. Transcribir en el Formulario X las conclusiones de estas verificaciones.

***Análisis**:*
En la mayoría de los casos la existencia de sobrantes debe considerarse como texto a ser retirado del componente. Sin embargo, debe prestarse atención que el sobrante sea alguna fracción que deba incluirse en otro componente especialmente en contexto y/o restricciones. Por otra parte en la mayoría de los casos los faltantes son irreparables, pero puede ocurrir que el mismo se infiera de los restantes componentes del escenario.

Forma IV. **COMPROBACIÓN SINTÁCTICA DE EPISODIOS DE UN ESCENARIO**

***Objetivo**:* Verificar si los episodios de un escenario se encuentran correctamente escritos.

***Pasos**:*

- Para cada uno de los episodios del escenario, comparar el texto del episodio, de las restricciones y de las excepciones con las sintaxis definidas para los mismos.
- Cuando exista algún error sintáctico:
  - ♦ Registrar en la columna FALTANTE toda omisión de elementos sintácticos obligatorios.
  - ♦ Registrar en la columna SOBRANTE todo elemento incluido en el componente que no se aparea con la sintaxis.
- Para cada componente verificar si los faltantes registrados son irreparables a partir de los restantes componentes del escenario; verificar también si los sobrantes pueden ser eliminados con razonable seguridad. Transcribir en el Formulario X las conclusiones de estas verificaciones.
- Tildar los casilleros en los que no se encontró ningún error.

***Análisis**:*
En la mayoría de los casos la existencia de sobrantes debe considerarse como texto a ser retirado del componente. Sin embargo, debe prestarse atención de que el sobrante sea alguna fracción que deba incluirse en otro componente especialmente en contexto y/o restricciones; eventualmente debe considerarse la posibilidad de que el sobrante requiera el agregado de un nuevo episodio. Por otra parte en la mayoría de los casos los faltantes son irreparables, pero puede ocurrir que el mismo se infiera de los restantes componentes del escenario.

Forma V. **COMPROBACIÓN PRAGMÁTICA DE EPISODIOS DE UN ESCENARIO**.

***Objetivo**:* Detectar episodios irrelevantes en el sentido que su presencia en el escenario posiblemente no sea necesaria.

***Pasos**:*

- Para cada uno de los episodios del escenario registrar en las columnas ACTORES, RECURSOS y SÍMBOLOS LEL la cantidad total mencionada en el episodio.
- Registrar en la columna DUDA todo episodio que no contenga ni actores ni recursos ni símbolos del LEL, excluyendo los episodios que son subescenarios.
- Para cada episodio en cuya columna de DUDA se ha registrado esta condición, verificar utilizando los restantes elementos del escenario si el mismo es irrelevante o si por el contrario debe ser modificado de tal manera de incluir algún actor o recurso o símbolo del LEL. Transcribir en el Formulario X las conclusiones de estas verificaciones.

***Análisis**:*

Los episodios de los escenarios deben referirse a actividades significativas de la situación descripta por el escenario y a su vez estas actividades razonablemente deben ser llevadas a cabo por actores que utilizan recursos generalmente incluidos en el LEL. Un episodio que no contenga ni actores, ni recursos ni símbolos del LEL debe ser estudiado a los efectos de dilucidar si se trata de un episodio innecesario en el escenario o si por el contrario si necesita ser modificado para incluir actores o recursos omitidos.

Forma VI. **CUMPLIMIENTO DEL LÉXICO EN UN ESCENARIO**.

Objetivo: Detectar usos incorrectos de símbolos del LEL, tales como uso con un significado diferente al registrado en el LEL, u omisión de destacar el símbolo.

Pasos:

- Registrar en la columna PALABRA / FRASE toda porción de texto destacada como símbolo del LEL, excluyendo los usos duplicados y los sinónimos.
- Para cada una de los presuntos símbolos del LEL comprobar su existencia en el mismo y registrar en la columna LEL a qué clase pertenece el símbolo involucrado. Cuando el símbolo no pertenezca al LEL indicar también este hecho en el Formulario X.
- Para todos los símbolos del LEL confirmados, inspeccionar el escenario buscando usos del mismo en los que se omitió destacarlos. Si se encontrara alguno registrar el componente en la columna SIN DESTACAR y en el Formulario X.
- Para todos los símbolos del LEL confirmados, inspeccionar el escenario buscando usos del mismo con un significado diferente al registrado en el LEL. Si se encontrara alguno, registrar el componente en la columna MAL USO y en el Formulario X.

Análisis:

Cuando un símbolo del LEL ha sido usado sin destacar y el uso que se dio al símbolo es compatible con el significado registrado en el LEL, la corrección del error simplemente consiste en escribirlo en forma destacada. Cuando el uso de un símbolo del LEL no es compatible con el significado registrado, la corrección del error consiste en reemplazar la mención del símbolo por sinónimos no incluidos en el LEL.

Forma VII. **CONTROL DE OCURRENCIA DE ACTORES EN UN ESCENARIO**.

***Objetivo**:* Verificar la coherencia de las personas o estructuras organizacionales que tienen un rol en el escenario, con los actores registrados en el componente correspondiente del mismo y/o que no son símbolos del LEL.

***Pasos**:*

- Para cada uno de los candidatos a actor[(1)], presentes en al menos un episodio, registrarlo en la columna CANDIDATO.

  [(1)] Personas o estructuras organizacionales que tienen un rol en el escenario.
- Agregar en la columna CANDIDATO aquellos que si bien están incluidos en el componente actores del escenario no han sido aún registrados en esta columna.
- Para cada uno de los candidatos registrar los lugares en los que se produce la ocurrencia: columnas TÍTULO, OBJETIVO, CONTEXTO o EPISODIOS, en este último caso indicar el número de los episodios en que aparece.
- Para cada uno de los candidatos comprobar si el mismo está incluido en el componente actores y registrar este hecho en la columna ACTOR. Cuando el candidato no está incluido en el componente actores registrarlo también en el Formulario X.
- Para cada uno de los candidatos, verificar que el mismo participe en algún episodio, en caso contrario registrar este hecho en el Formulario X.
- Para cada uno de los candidatos registrados verificar si pertenece al LEL y registrar en la columna LEL la clase a la que pertenece el símbolo involucrado. Para realizar este paso puede utilizarse el formulario VI. Cuando el símbolo no pertenezca al LEL o perteneciendo no corresponde a la clase sujeto, indicarlo también en el formulario X.

***Análisis**:*

Cuando el CANDIDATO no está registrado en el componente actores pero es un símbolo del LEL perteneciente a la clase sujeto y participa en al menos un episodio, la corrección es simple ya que sólo consiste en la inclusión del mismo en el componente actores. Por otra parte cuando el CANDIDATO no pertenece a la clase sujeto, esté o no incluido en el componente actores, la corrección es de mayor importancia ya que se debe analizar también el LEL para determinar si corresponde realizar una corrección en el mismo. Cuando un candidato está registrado en el componente actores del escenario pero no participa en ningún episodio debe detectarse si se trata de una inclusión innecesaria o si por el contrario se omitió algún episodio.

Forma VIII. **CONTROL DEL USO DE RECURSOS EN UN ESCENARIO**.

***Objetivo**:* Verificar la coherencia de medios de soporte, dispositivos u otros elementos pasivos necesarios para estar disponibles en el escenario, con los recursos registrados en el componente correspondiente del mismo y/o que no son símbolos del LEL.

***Pasos**:*

- Para cada uno de los candidatos a recurso(2), presentes en al menos un episodio, registrarlos en la columna CANDIDATO.

  (2) Medios de soporte, dispositivos u otros elementos pasivos necesarios para estar disponibles en el escenario.
- Agregar en la columna CANDIDATO aquellos que si bien están incluidos en el componente recurso del escenario no han sido aún registrados en esta columna.
- Para cada uno de los candidatos registrar los lugares en los que se produce la ocurrencia: columnas TÍTULO, OBJETIVO, CONTEXTO o EPISODIOS, en este último caso agregar el número de los episodios en que aparece
- Para cada uno de los candidatos registrados comprobar si el mismo está incluido en el componente recursos y registrar este hecho en la columna RECURSO. Cuando el candidato no está incluido en el componente recursos registrar este hecho también en el Formulario X.
- Para cada uno de los candidatos, verificar que el mismo participe en algún episodio, en caso contrario registrar este hecho en el Formulario X.
- Para cada uno de los candidatos registrados verificar si pertenece al LEL y registrar en la columna LEL la clase a la que pertenece el símbolo involucrado. Para realizar este paso puede utilizarse el formulario VI. Cuando el símbolo no pertenezca al LEL o perteneciendo no corresponde a la clase objeto, indicar también este hecho en el Formulario X.

***Análisis**:*

Cuando el CANDIDATO no está registrado en el componente recursos pero es un símbolo del LEL perteneciente a la clase objeto y participa en al menos un episodio, la corrección es simple ya que sólo consiste en la inclusión del mismo en el componente recursos. Por otra parte cuando el CANDIDATO no pertenece a la clase objeto, esté o no incluido en el componente recursos, la corrección es de mayor importancia ya que se debe analizar también el LEL para determinar si corresponde realizar una corrección en el mismo. Cuando un candidato está registrado en el componente recursos del escenario pero no participa en ningún episodio debe detectarse si se trata de una inclusión innecesaria o si por el contrario se omitió algún episodio.

Forma IX. **CONTROL SINTÁCTICO DE TIPOS DE EPISODIOS EN UN ESCENARIO**

***Objetivo**:* Detectar usos incorrectos de los indicadores de episodios opcionales, no secuenciales y condicionales.

***Pasos**:*

- Registrar en las columnas CANT y EPISODIOS del cuadro ubicado en la parte superior del formulario, la cantidad y en qué episodios existen indicadores opcionales que abren (**[**), que cierran (**]**) e indicadores de no secuencia (#), tanto en el comienzo, en el interior como en el final de los mismos. También registrar el total.
- Registrar en las columnas CANT y EPISODIOS del cuadro ubicado en la parte superior del formulario, la cantidad y en qué episodios existen indicadores condicionales SI y ENTONCES tanto en el comienzo, en el interior como en el final de los mismos. También registrar el total.
- Verificar la existencia de indicadores opcionales que abren (**[**) ubicados al principio de un episodio, precedidos por otro indicador igual sin que exista el correspondiente indicador que cierra (**]**) al final de algún episodio previo. Registrar la cantidad y la lista de episodios en que esto ocurre en el cuadro ubicado en la parte inferior del formulario.
- Verificar la existencia de indicadores opcionales que cierran (**]**) ubicados al final de un episodio, no precedidos por un indicador opcional que abre (**[**) al principio del episodio o de alguno previo. Registrar la cantidad y la lista de episodios en que esto ocurre en el cuadro ubicado en la parte inferior del formulario.
- Verificar la existencia de indicadores de no secuencia (#) ubicados al principio del episodio, precedidos por otro indicador de no secuencia (#), también ubicado al principio de un episodio y no precedidos por un indicador de no secuencia (#) ubicado al final de algún episodio intermedio. Registrar la cantidad y la lista de episodios en que esto ocurre en el cuadro ubicado en la parte inferior del formulario.
- Verificar la existencia de indicadores de no secuencia (#) ubicados al final de un episodio, no precedidos por un indicador de no secuencia (#) al principio de algún episodio previo. Registrar la cantidad y la lista de episodios en que esto ocurre en el cuadro ubicado en la parte inferior del formulario.
- Verificar la existencia de indicadores de no secuencia (#) conteniendo un solo episodio. Registrar la cantidad y la lista de episodios en que esto ocurre en el cuadro ubicado en la parte inferior del formulario.

***Análisis**:*

Los indicadores opcionales que abren (**[**) y que cierran (**]**) sólo pueden estar presentes en el comienzo y al final de los episodios respectivamente, los indicadores de opcionalidad sólo pueden estar al comienzo y los indicadores de no secuencia (#) pueden estar al comienzo y/o al final, por lo que la presencia de cualquiera de ellos en el interior de un episodio es un error. En cambio el indicador ENTONCES sólo puede estar en el interior de un episodio. Por otra parte, la condición de opcionalidad y no secuencia no pueden anidarse en si mismas, por lo que toda ocurrencia de un indicador en el comienzo de un episodio debe estar precedida por el cierre del indicador anterior excepto en la primera ocurrencia. El indicador de no secuencia debe encerrar al menos dos episodios pues de lo contrario carecería de sentido.

La cantidad de indicadores opcionales que abren (**[**) y que cierran (**]**) deben ser iguales, la cantidad de indicadores de no secuencia (#) debe ser par y finalmente la cantidad de indicadores SI a la cantidad de episodios involucrados.

Si una sentencia condicional involucra más de un episodio, la misma puede meramente indicar la necesidad de desdoblar la sentencia condicional en varias o puede requerir desdoblar todo el escenario.

F.2.2. Formularios de Intra Inspección

TABLA 1 **Forma I**

REPORTE DE CONSISTENCIA INTERNA DE ESCENARIOS

Proyecto: .....

Inspector: .....

Fecha de Verificación: .....

Esfuerzo de Verificación: .....

Autor(es): .....

Versión: .....

Fecha de construcción: .....

Esfuerzo de construcción: .....

Versión del LEL utilizada: .....

TABLA 1 **Forma II.--------**

NºEsc.

RESUMEN CUANTITATIVO DE UN ESCENARIO

.....

Nombre Escenario

RUBRO	RELACIÓN[+]	CANTIDAD
OBJETIVO		
CONTEXTO		
UBICACIÓN GEOGRÁFICA		
UBICACIÓN TEMPORAL		
PRECONDICIONES		
ACTORES		
RECURSOS		
EPISODIOS		
SECUENCIALES		
NO SECUENCIALES		
SUBESCENARIOS		
CONDICIONALES		
OPTATIVOS		
RESTRICCIONES		
EXCEPCIONES		
REFERENCIAS TOTALES AL LEL		
REFERENCIAS ÚNICAS AL LEL[*]		

[*] Sin incluir los usos duplicados

[+] Si o No. Relación con el título del escenario

TABLA 1 **Forma III.--------**

NºEsc.

COMPROBACIÓN SINTÁCTICA DE UN ESCENARIO

COMPONENTE	SINTAXIS	**FALTANTE**	**SOBRANTE**
Título	*Frase | ([Actor | Recurso] + Verbo + Predicado)*		
Objetivo	*[Actor | Recurso] + Verbo +*		
Contexto			
Ubic. Geográfica	*Frase | Nombre*		
Ubic. Temporal	*Frase | Nombre*		
Precondiciones	*[Sujeto | Actor | Recurso] + Verbo + Predicado*		
Actores	*Nombre*		
Recursos	*Nombre*		
Excepciones	*Causa (Solución)* Causa: *(Frase | ([Sujeto | Actor | Recurso] + Verbo + Predicado)) + {Nro. Episodio}* Solución: *Título | ([Sujeto | Actor | Recurso] + Verbo + Predicado)*		

TABLA 1 **Forma IV.--------.--------**

NºEsc. NºSec.

COMPROBACIÓN SINTÁCTICA DE EPISODIOS DE UN ESCENARIO

	EPISODIO ([Actor] + Verbo + Predicado) | ([Actor] + [Verbo] +Título)		*RESTRICCIÓN* ([Sujeto | Actor | Recurso] + [**No**] **Debe** + Verbo + Predicado) | Frase	
NRO[$]	FALTANTE	SOBRANTE	FALTANTE	SOBRANTE

$ Número de episodio con error sintáctico

TABLA 1 **Forma V.--------.--------**
NºEsc. NºSec.

COMPROBACIÓN PRAGMÁTICA DE EPISODIOS DE UN ESCENARIO

NRO[$]	ACTORES[o]	RECURSOS[o]	SÍMBOLOS LEL[o]	DUDA[+]
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

[$] Número de episodio
[o] Totales
[+] Marcar en caso de episodios sin actores ni recursos ni símbolos del LEL

TABLA 1 **Forma VI.--------.--------**

NºEsc. NºSec.

CUMPLIMIENTO DEL LÉXICO

EN UN ESCENARIO

	PALABRAS/FRASES*º	LEL+	SIN DESTACAR%	MAL USO%
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				

* Sin incluir los usos duplicados
% Indicar en qué componente
º Registrar toda palabra o frase destacada
+ Sujeto, Objeto, Verbo, Estado, Otra Clase, No existe en LEL

TABLA 1 **Forma VII.--------.--------**

NºEsc. NºSec.

CONTROL DE OCURRENCIA DE ACTORES EN UN ESCENARIO

	OCURRENCIA				EXISTENCIA	
CANDIDATO[%] [*]	TÍTULO[o]	OBJETIVO[o]	CONTEXTO[o]	EPISODIOS[^]	ACTOR[$]	LEL[+]

[%] Candidato a actor: persona o estructura organizacional que tiene un rol en el escenario [o] Totales

[$] Si o No. Perteneciente al componente Actores [^] Lista de episodios, incluyendo restricciones y excepciones

[+] Sujeto, Objeto, Verbo, Estado, Otra Clasificación, No existe en LEL [*] Sin incluir los usos duplicados

TABLA 1 **Forma VIII.--------.--------**

NºEsc. NºSec.

CONTROL DEL USO DE RECURSOS EN UN ESCENARIO

	OCURRENCIA				EXISTENCIA	
CANDIDATO[%,*]	TÍTULO[o]	OBJETIVO[o]	CONTEXTO[o]	EPISODIOS[^]	RECURSO[$]	LEL[+]

[%] Candidato a recurso: medios de soporte u otros elementos pasivos necesarios para estar disponibles en el escenario
[$] Si o No. Perteneciente a componente recursos [o] Totales [^] Lista de episodios, incluyendo restricciones y excepciones
[+] Sujeto, Objeto, Verbo, Estado, Otra Clasificación, No existe en LEL [*] Sin incluir usos duplicados

TABLA 1 **Forma IX.--------**
NºEsc

CONTROL SINTÁCTICO DE TIPOS DE EPISODIOS EN UN ESCENARIO

	COMIENZO		INTERIOR		FINAL		TOTAL
INDICADORES	CANT[o]	EPISODIOS[*]	CANT[o]	EPISODIOS[*]	CANT[o]	EPISODIOS[*]	CANT[o]
Opcionales que abren ([ )							
Opcionales que cierran ( ])							
No secuencia (#)							
Indicadores SI							
Indicadores ENTONCES							

RELACIONES ENTRE INDICADORES	CANTIDAD[o]	EPISODIOS[*]
Indicadores ([ ) al principio del episodio sin cierre previo[+]		
Indicadores ( ]) al final del episodio sin apertura previa		
Indicadores (#) al principio del episodio sin cierre previo[+]		
Indicadores (#) al final del episodio sin apertura previa		
Indicadores (#) encerrando un solo episodio		
Sentencias condicionales con más de un episodio		

[o] Totales [*] Lista de episodios [+] Excepto en la primera aparición

TABLA 1 **Forma X.--------.--------**

NºEsc. NºSec.

LISTA CANDIDATA DE DEOs
OBSERVACIONES

Correcciones propuestas			
Nro.Forma	Nro.Sec.*	Ítem	Observación

Información a recabar			
Nro.Forma	Nro.Sec.*	Ítem	Observación

- Si corresponde

TABLA 1 **Forma XI.--------**
NºSec.

COMENTARIO GENERAL

F.3. Inter Inspección

F.3.1. Guías de Instrucción Inter Inspección

Tabla 2. **REPORTE DE CONSISTENCIA ENTRE ESCENARIOS.**
***Objetivo**:* Detectar discrepancias, errores y omisiones entre el conjunto de escenarios de un punto de vista.
***Precondiciones**:* • El responsable de la verificación debe conocer el LEL. • Esta verificación se debe realizar después de la Verificación de Consistencia interna de Escenarios. • El soporte de los escenarios debe permitir la inserción de información.
***Proceso General**:* • Completar los formularios I a XI en el orden de su numeración y de acuerdo a las indicaciones dadas para cada una. • En el caso de los formularios III a XIII, utilizar tantas hojas como sea necesario numerándolas correlativamente. • Una vez completado cada formulario o simultáneamente al llenado del mismo, realizar un análisis de los datos obtenidos. • Registrar en el formulario XII, en el cuadro CORRECCIONES PROPUESTAS todas las recomendaciones posibles orientadas a mejorar la descripción de los escenarios, y en el cuadro DUDAS toda cuestión no discernible y que requiera la validación con el usuario o con el autor del escenario. En ambos casos, hacer referencia al formulario que originó la observación. • Finalizada la verificación, revisar los comentarios registrados en los formularios XII y realizar en el formulario XIII un comentario general, basado en el nivel de correcciones y de dudas detectados, sobre la calidad del conjunto de escenarios y sobre todo otro aspecto que corresponda y que permita tomar decisiones sobre los pasos a seguir.
Análisis: En los formularios II a XII es posible detectar inconvenientes de diferente naturaleza en el conjunto de escenarios bajo estudio. En algunos casos, estos inconvenientes son subsanables en el sentido que con la información disponible es posible discernir con seguridad la corrección a realizar; mientras que en otros es necesario recurrir al autor/es de los mismos o eventualmente recabar información del Universo de Discurso. Todo problema debe ser catalogado como subsanable sólo si existe una razonable confianza en la corrección que se propone, en caso contrario se la debe considerar no subsanable.

Forma I. **CARÁTULA**.
***Objetivo**:* Identificar el proyecto y el conjunto de escenarios del punto de vista a consistir.
***Pasos**:* • Registrar la información básica y disponible del proyecto y del conjunto de escenarios. • Al finalizar el proceso, registrar el esfuerzo de verificación del conjunto de escenarios.

***Análisis**:*
No corresponde.

Forma II. **RESUMEN CUANTITATIVO DEL CONJUNTO DE ESCENARIOS**.

***Objetivo**:* Brindar una idea general del volumen de información presente en el conjunto de escenarios.

***Pasos**:*

- Clasificar los escenarios según su nivel de jerarquía en Escenarios Integradores(1), Escenarios(2) y Sub-escenarios(3).

  (1) Escenario Integrador: aquél que representa una visión global de la totalidad o parte del problema. Sus episodios son Escenarios.

  (2) Escenario: aquél que representa una situación particular y se encuentra en el primer nivel de la jerarquía. Puede contener o no Sub-escenarios.

  (3) Sub-escenario: aquél que es a su vez un episodio o una excepción de otro Escenario o Sub-escenario.
- Registrar en la columna CANTIDAD los totales para cada rubro sin incluir los usos repetidos, salvo el caso de episodios. En el caso de restricciones, incluir aquellas adjuntas al contexto, a los recursos y a los episodios. Utilizar el LEL para contabilizar el total de símbolos y los símbolos que pertenecen a la clase Sujeto y a la clase Objeto. No contabilizar el total de Actores y Recursos; al completar los formularios VIII y IX trasladar dichos totales a este formulario.

***Análisis**:*
En el caso de una baja cantidad de Sub-escenarios, evaluar el hecho como posible indicador de descripciones de escenarios excesivamente planas.
En el caso de un valor bajo en el promedio de episodios por escenario, considerar la posible existencia de una alta disgregación de situaciones.
En el caso de un valor alto en el promedio de episodios por escenario, considerar la posible existencia de una excesiva concentración de situaciones en un sólo escenario.
En el caso de valores bajos en excepciones y restricciones, considerar la posibilidad de omisiones.
En el caso en que el uso del vocabulario del LEL deje una cantidad significativa sin mencionar, considerar la posibilidad de omisión de detalles y/o situaciones.
En el caso que la cantidad de sujetos del LEL sea significativamente mayor que la cantidad de actores, considerar la posibilidad de omisión de situaciones o de no uso del vocabulario del LEL. En el caso inverso, considerar la posibilidad de estar involucrando situaciones fuera del alcance del problema o de la incompletitud del LEL.
En el caso que la cantidad de objetos del LEL sea significativamente mayor que la cantidad de recursos, considerar la posibilidad de omisión de información o de no uso del vocabulario del LEL. En el caso inverso, considerar la posibilidad del estudio de acciones fuera del alcance del problema o de un excesivo nivel de detalle o de la incompletitud del LEL.

Forma III. **CONTROL CRUZADO DE ESCENARIOS INTEGRADORES.**
***Objetivo**:* Asegurarse que los escenarios identificados como integradores estén correctamente clasificados.
***Pasos**:* • Confeccionar una lista con todos los escenarios identificados como Escenarios Integradores y registrarlo en la columna ESCENARIO INTEGRADOR, ordenados por número de escenario. • Registrar en la columna TOTAL DE EPISODIOS la cantidad total de episodios que componen cada Escenario Integrador. • Registrar en la columna ESCENARIOS la lista de episodios identificados como Escenarios (en letra mayúscula y pertenecientes a la clasificación Escenario) que componen el Escenario Integrador. • Para cada Escenario Integrador, registrar en la columna SUB-ESCENARIOS la lista de episodios identificados como Sub-escenarios (en letra mayúscula y pertenecientes a la clasificación Sub-escenario) y transcribirla también al formulario XII con las observaciones que correspondan. • Para cada Escenario Integrador, registrar en la columna SIMPLES la lista de episodios que representan acciones simples y transcribirla también al formulario XII con las observaciones que correspondan.
***Análisis**:* Para cada Escenario Integrador, examinar todos los episodios simples existentes con el fin de determinar si se trata de uno de cuatro casos siguientes: a) episodios que no han sido destacados como escenarios, o b) un escenario mal clasificado como Escenario Integrador, o c) acciones que manifiestan la existencia de situaciones total o parcialmente ignoradas en el conjunto de escenarios, o d) acciones incluidas innecesariamente en el Escenario Integrador por ya existir en algún Escenario o Sub-escenario. Si no se puede asegurar ninguna de estos casos, entonces proponer una revisión exhaustiva del Escenario Integrador y de los episodios involucrados. Si se puede asegurar que se trata de un episodio que no ha sido destacado como un Escenario, la corrección del error es simple ya que sólo consiste en destacar el episodio como Escenario. Si se puede asegurar que se trata de un episodio que no ha sido destacado como un Sub-escenario, entonces no alcanza con sólo destacar el episodio como Sub-escenario ya que éstos no deben existir en un Escenario Integrador y, por lo tanto, analizar la redefinición del Escenario Integrador o del Sub-escenario. Si se puede asegurar que se trata de un escenario mal clasificado, reclasificar el Escenario Integrador como Escenario y en caso que la columna Escenarios no esté vacía, reclasificar cada Escenario como Sub-escenario. Si se puede asegurar que se trata de acciones total o parcialmente ignoradas, recomendar el agregado de los escenarios necesarios o la inclusión de dichas acciones en los escenarios correspondientes. Si se puede asegurar que se trata de acciones incluidas innecesariamente en el Escenario Integrador, recomendar la eliminación de estos episodios. Si el Escenario es Integrador y existen Sub-escenarios, la corrección del error es importante ya que requiere redefinir el Escenario Integrador o los Sub-escenarios incluidos.

Forma IV. **CONTROL CRUZADO DE ESCENARIOS Y SUB-ESCENARIOS**.
***Objetivo**:* Detectar la inexistencia de sub-escenarios mencionados como tales en episodios o excepciones y detectar sub-escenarios existentes no destacados en episodios o excepciones.
***Pasos**:* • Registrar en la columna CANDIDATO A SUB-ESCENARIO aquellos episodios y excepciones destacados como Sub-escenarios dentro de Escenarios y otros Sub-escenarios, sin incluir usos duplicados. • Verificar la existencia del candidato a sub-escenario dentro del conjunto de escenarios y registrarlo en la columna EXISTE. Si no existe el sub-escenario, registrar este hecho en el formulario XII. • Agregar en la columna CANDIDATO A SUB-ESCENARIO la lista de Sub-escenarios, clasificados en el formulario II y que no han sido registrados en el primer paso, y confirmar su existencia en la columna EXISTE. • Para cada candidato a sub-escenario, registrar en la columna ESCENARIO los escenarios que lo mencionan en sus episodios y excepciones. Si no se encontrara ninguna ocurrencia, registrar este hecho en el formulario XII. • Registrar en la columna NO DESTACADO la lista de escenarios /episodios / excepciones donde los candidatos a sub-escenarios no se encuentran destacados e informarlo también en el formulario XII para su corrección.
***Análisis**:* Para aquellos candidatos a sub-escenarios no encontrados, analizar su mención en los escenarios para poder determinar si se trata de una acción simple o de una situación que involucra más de una acción y que, por lo tanto, requiere la existencia de un escenario. De ser necesario, analizar también los episodios que lo anteceden y lo preceden. En el caso de sub-escenarios con ocurrencia nula en episodios y excepciones, analizar la posibilidad que se trate de un escenario mal clasificado. Para aquellos candidatos a sub-escenarios confirmados (encontrados) y que no han sido destacados en algún escenario, la corrección es simple ya que consiste en destacar toda ocurrencia del mismo.

Forma V. **CONTROL CRUZADO DE PRECONDICIONES EN EL CONTEXTO**.
***Objetivo**:* Detectar errores o discrepancias en las precondiciones de los escenarios u omisión de información para satisfacer dichas precondiciones.

***Pasos**:*

- Registrar en la columna ESCENARIO la lista de escenarios que presenten al menos una precondición, ordenada por número de escenario.
- Completar en la columna PRECONDICION un renglón por cada precondición del escenario.
- Clasificar el origen de las precondiciones en internas(1) o externas(2) y transcribirlo a la columna ORIGEN. En caso de no poder catalogarse el origen, completar con la palabra *"Error"* y registrarlo también en el formulario XII.
  (1) Precondición Interna: aquella precondición satisfecha por algún escenario.
  (2) Precondición Externa: aquella precondición satisfecha fuera del conjunto de escenarios.
- Registrar en la columna QUIEN LA SATISFACE la lista de escenarios que satisfacen las precondiciones internas, indicando de ser posible el o los números de episodios involucrados. En caso de no poder establecerse esto, registrarlo en el formulario XII con las observaciones que correspondan.

***Análisis**:*

En el caso de precondiciones con error en el origen, considerar la posibilidad de corrección de la precondición o su eliminación.

En el caso que una precondición interna no sea satisfecha por al menos un escenario, considerar la posibilidad de corregir la precondición o analizar la falta de un escenario o la falta de episodios dentro de escenarios existentes, para satisfacer dicha precondición interna.

Forma VI. **CONTROL DE PRECONDICIONES DE SUB-ESCENARIOS**.

***Objetivo**:* Detectar errores en las precondiciones de sub-escenarios u omisión de información en los escenarios que contienen sub-escenarios, considerando los vínculos jerárquicos Escenario – Sub-escenario y Escenario Integrador – Escenario.

***Pasos**:*

- Confeccionar una lista con cada par Escenario – Sub-escenario y registrarlo en las columnas ESCENARIO y SUB-ESCENARIO respectivamente, ordenados por número de escenario. En el caso de presentar el Sub-escenario precondiciones registradas como internas en el formulario V, anotar en la columna PRECONDICIÓN SUB-ESCENARIO en un renglón separado cada una. En el caso de no presentar precondiciones internas, registrar este hecho en el formulario XII.
- Agregar a la lista anterior todos los pares Escenario Integrador – Escenario y tratarlos en lo sucesivo como si fueran Escenario – Sub-escenario respectivamente. Registrar para estos pares la misma información indicada en el paso previo.
- Establecer la relación existente entre la precondición del Sub-escenario y la precondición del Escenario y/o los episodios anteriores a su mención en el Escenario, y registrarlo en la columna RELACIÓN CON EL ESCENARIO. De no poder establecerse esta relación, marcarlo en la columna OBS y registrarlo en el formulario XII con las observaciones que correspondan.

***Análisis**:*
En el caso que el Sub-escenario no presente ninguna precondición, considerar la posible omisión de la misma. Para ello evaluar uno de los siguientes casos:
a) Sub-escenario que no es primer episodio del Escenario, o
b) Sub-escenario que es primer episodio del Escenario.

Si el Sub-escenario no es el primer episodio del Escenario, analizar la secuencia de los episodios anteriores en el Escenario. Si esta secuencia es mandatoria, fabrica naturalmente la precondición del Sub-escenario. En el caso contrario analizar el caso b).
Si el Sub-escenario es el primer episodio del Escenario y éste presenta precondición, analizar dicha precondición para evaluar si no es también la precondición del Sub-escenario.
En el caso que la precondición del Sub-escenario no sea satisfecha por al menos un episodio del Escenario o que no coincida con la precondición del Escenario, evaluar uno de estos casos:
a) error u omisión en la precondición del Sub-escenario, o
b) error u omisión en la precondición del Escenario, o
c) omisión de información o episodios en el Escenario para satisfacer dicha precondición.

Si no se puede asegurar ninguno de estos casos, entonces recomendar una revisión general del Escenario y del Sub-escenario.
Si se puede asegurar que se trata del caso a) o del b) sugerir la corrección de la precondición del Sub-escenario o del Escenario respectivamente o ambas precondiciones.
Si se puede asegurar que se trata del caso c) considerar la posibilidad de incluir en el Escenario los episodios faltantes o la información omitida en episodios existentes.

Forma VII. **CONTROL DE SUPERPOSICIÓN DE ESCENARIOS**.
***Objetivo**:* Detectar la superposición de información entre escenarios.

***Pasos**:*

- Confeccionar una lista de pares de escenarios todos contra todos, excluyendo los Escenarios Integradores y resgistrarlos en las columnas ESCEN**i** y ESCEN**j** respectivamente, ordenados por número de escenario.
- Para cada par, registrar en la columna $A_{\cup ij}$ la unión de Actores, en la columna $A_{\cap ij}$ el total de coincidencias de Actores, en la columna $R_{\cup ij}$ la unión de Recursos, en la columna $R_{\cap ij}$ el total de coincidencias de Recursos, en la columna $C_{\cup ij}$ la unión de Contextos y en la columna $C_{\cap ij}$ el total de coincidencias de Contextos.
- Para cada par de escenarios, calcular el Índice de cercanía entre el escenario **i** y el escenario **j** como:

$$I_{ij} = (\alpha * C_{\cap ij} + \beta * A_{\cap ij} + \gamma * R_{\cap ij}) / (\alpha * C_{\cup ij} + \beta * A_{\cup ij} + \gamma * R_{\cup ij})$$

donde:

$C_{\cap ij} = |\ Cont(E_i) \cap Cont(E_j)\ |$, $A_{\cap ij} = |\ Act(E_i) \cap Act(E_j)\ |$, $R_{\cap ij} = |\ Rec(E_i) \cap Rec(E_j)\ |$,
$C_{\cup ij} = |\ Cont(E_i) \cup Cont(E_j)\ |$, $A_{\cup ij} = |\ Act(E_i) \cup Act(E_j)\ |$, $R_{\cup ij} = |\ Rec(E_i) \cup Rec(E_j)\ |$,
α, β, γ, factores de peso,
$Cont(E_k)$ = Conjunto de enunciados del contexto del escenario k ,
$Act(E_k)$ = Conjunto de actores del escenario k,
$Rec(E_k)$ = Conjunto de recursos del escenario k,
$i, j \in N \wedge i < j$,
N= Conjunto de números naturales.

- Para cada par cuyo Índice supere un determinado valor, registrar en la columna OBJETIVO el total de coincidencias en el Objetivo (considerar partes comunes) y en la columna EPISODIO el total de coincidencias en los Episodios.
- Para los pares considerados en el paso anterior, marcar en la columna OVERLAP aquellos que presenten un alto grado de coincidencia y registrar estos pares en el formulario XII.

***Análisis**:*

El índice es útil para reducir el esfuerzo en la cantidad de comparaciones, ya que sólo se examinan los pares que podrían tener superposición.

Si un par de escenarios tiene coincidencia total o parcial en sus objetivos, existe una superposición entre ellos. Esta superposición se podrá corregir, eliminando completamente un escenario o redefiniendo los objetivos de uno o ambos escenarios y los restantes componentes.

Si un par de escenarios tiene objetivos disjuntos pero tienen episodios comunes, esto puede deberse a la inclusión indebida de estos episodios en uno de los escenarios. La verosimilitud de esta suposición crece con la cantidad de episodios comunes.

Forma VIII. **CONTROL DE OCURRENCIA DE ACTORES**.

***Objetivo**:* Detectar actores no incluidos en el LEL y con gran participación en los escenarios.

***Pasos**:*

- Armar una lista con los actores que participan en el conjunto de escenarios, sin incluir los usos duplicados y registrarlos en la columna ACTOR.
- Para cada actor, registrar en las columnas ESCENARIOS y EPISODIOS el total de escenarios y el total de episodios respectivamente en los que participa y registrar en la columna LEL si pertenece al LEL.
- Para los actores que no pertenezcan al LEL, registrarlos en el formulario XII, indicando si tienen mucha o poca participación en los escenarios y episodios, con las observaciones que correspondan.

***Análisis**:*

En el caso de Actores con una alta participación en el total de escenarios y en el total de episodios y que no son símbolos del LEL, considerar la posibilidad de omisión del actor como símbolo en el LEL.

En el caso de actores con baja participación en escenarios y que no pertenecen al LEL, analizar los episodios en los que actúan y considerar la posibilidad que dichos episodios estén fuera del alcance de la aplicación.

Si el actor es un símbolo del LEL no se verifica su pertenencia a la clase Sujeto, ya que esta verificación es realizada en la Tabla 1.

Forma IX. **CONTROL DE USO DE RECURSOS**.

***Objetivo**:* Detectar recursos no incluidos en el LEL y con gran participación en los escenarios.

Pasos:

- Armar una lista con los recursos empleados en el conjunto de escenarios, sin incluir los usos duplicados y registrarlos en la columna RECURSO.
- Para cada recurso, registrar en las columnas ESCENARIOS y EPISODIOS el total de escenarios y el total de episodios respectivamente en los que se menciona y registrar en la columna LEL si pertenece al LEL.
- Para los recursos que no pertenezcan al LEL, registrarlos en el formulario XII, indicando si tienen mucha o poca participación en los escenarios y episodios, con las observaciones que correspondan.

***Análisis**:*

En el caso de Recursos con una alta participación en el total de escenarios y en el total de episodios y que no son símbolos del LEL, considerar la posibilidad de omisión del recurso como símbolo en el LEL.

En el caso de recursos con baja participación en escenarios y que no pertenecen al LEL, analizar los episodios en los que participan y considerar la posibilidad que dichos episodios estén fuera del alcance de la aplicación o que haya un excesivo detalle de información.

Si el recurso es un símbolo del LEL no se verifica su pertenencia a la clase Objeto, ya que esta verificación es realizada en la Tabla 1.

Forma X. **SÍMBOLOS DEL LEL NO UTILIZADOS**.
***Objetivo**:* Detectar omisiones en el conjunto de escenarios mediante el análisis del grado de cubrimiento de los símbolos del LEL.
***Pasos**:* • Registrar en la columna SÍMBOLO aquellos símbolos del LEL no empleados en ningún escenario, preservando el orden establecido en el LEL. • Registrar en la columna LEL la clase a la que pertenece el símbolo. • Contabilizar el total de impactos que presenta y registrarlo en la columna IMPACTOS. • Para aquellos símbolos con más de un impacto, contabilizar en la columna REF el total de referencias a otros símbolos que se mencionan en sus impactos, sin incluir los usos duplicados y registrarlo en el formulario XII.
***Análisis**:* En el caso de ser alta la cantidad de símbolos del LEL no utilizados, considerar la posibilidad de haber hecho un escaso uso del LEL en la descripción de los escenarios. En el caso de Sujetos en el LEL no utilizados, considerar la posibilidad de escenarios omitidos o incompletos. En el caso de Objetos en el LEL no utilizados y que hacen mucha referencia a otros símbolos, considerar la posibilidad de escenarios incompletos. En el caso de Verbos en el LEL no utilizados y que hacen mucha referencia a otros símbolos, considerar la posibilidad de escenarios omitidos o incompletos. En el caso de Estados en el LEL no utilizados y que hacen mucha referencia a otros símbolos, considerar la posibilidad de acciones omitidas o incompletas.

Forma XI. **CONTROL DE IMPACTOS VERSUS ESCENARIOS / EPISODIOS**.
***Objetivo**:* Detectar situaciones omitidas en los escenarios y/o discrepancias entre los sujetos y los actores de determinadas acciones.
***Pasos**:* • Numerar correlativamente en el LEL los impactos de cada símbolo perteneciente a la clase Sujeto. • Completar la columna SUJETO con cada símbolo del LEL perteneciente a la clase Sujeto que presente al menos un impacto, preservando el orden establecido en el LEL. • Para cada símbolo, registrar en un renglón separado en la columna IMPACTO cada número de impacto. • Identificar los escenarios que satisfacen cada impacto y especificar en la columna ESCENARIOS / EPISODIOS la lista de escenarios y de episodios involucrados y en la columna ACTORES la lista de los actores que participan en dichas acciones. • Si un impacto no es satisfecho por ningún escenario o es satisfecho parcialmente o el sujeto del impacto no coincide con los actores del escenario, marcarlo en la columna OBS y registrarlo en el formulario XII.

***Análisis**:*
En el caso que un impacto no sea satisfecho por al menos un escenario, considerar la posibilidad de la omisión de una situación.
En el caso que el sujeto del impacto no coincida con ningún actor del escenario, considerar la posibilidad de una discrepancia entre quienes ejecutan las acciones.
En el caso de un impacto satisfecho parcialmente, considerar la posibilidad de omisión de episodios o de detalles en episodios existentes.

F.3.2. Formularios de Inter Inspección

TABLA 2 **Forma I**

REPORTE DE CONSISTENCIA ENTRE ESCENARIOS

Proyecto:.....

Escenarios inspeccionados

Autor(es): …….....……

Versión: .....……..

Fecha de construcción: .....……..

Técnica de construcción: .....……..

Esfuerzo de construcción: .....……..

LEL utilizado

Versión: .....…….....

Datos Generales de la inspección

Inspector: ….....….....

Fecha de Verificación: ….....……..

Esfuerzo de Verificación: …..

TABLA 2 **Forma II**

RESUMEN CUANTITATIVO DEL CONJUNTO DE ESCENARIOS

	CANTIDAD
ESCENARIOS	
ESCENARIOS INTEGRADORES	
ESCENARIOS	
SUBESCENARIOS	
ACTORES*	
RECURSOS*	
EPISODIOS+	
EXCEPCIONES*	
RESTRICCIONES*	
REFERENCIAS UNICAS DEL LEL*	
SÍMBOLOS TOTALES DEL LEL	
SÍMBOLOS DEL LEL SUJETO	
SÍMBOLOS DEL LEL OBJETO	

+ Incluyendo los episodios que son subescenarios

* Sin incluir los usos duplicados

TABLA 2 **Forma III.--------**

NºSec.

CONTROL CRUZADO DE ESCENARIOS INTEGRADORES

ESCENARIO INTEGRADOR[@]	TOTAL DE EPISODIOS	LISTA DE EPISODIOS		
		ESCENARIOS[o]	SUBESCENARIOS[o]	SIMPLES[o]

[@] Número de escenario [o] Lista de episodios

TABLA 2 **Forma IV.--------**

NºSec.

CONTROL CRUZADO DE ESCENARIOS Y SUBESCENARIOS

CANDIDATO A SUBESCENARIO*	EXISTE+	ESCE-NARIO%	NO$ DESTACADO

***** Sin incluir los usos duplicados + Existe el escenario (Sí/No)

% Lista de escenarios que lo mencionan como episodio o excepción

$ Lista de escenarios / episodios / excepciones donde no está destacado

TABLA 2 **Forma V.--------**

NºSec.

CONTROL CRUZADO DE PRECONDICIONES EN EL CONTEXTO

ESCENARIO[@]	**PRECONDICIÓN**	ORIGEN[$]	**QUIEN LA SATISFACE[*]**

[@] Número de escenario

[$] Origen de Precondición: Interna, Externa, Error * Lista de escenarios, completar cuando Origen = I

TABLA 2 **Forma VI.--------**

NºSec.

CONTROL DE PRECONDICIONES DE SUB-ESCENARIOS

ESCENARIO@	SUBESCEN.@	**PRECONDICIÓN SUBESCENARIO**	**RELACIÓN CON EL ESCENARIO$**	**OBS***

@ Número de escenario $ Precondición y/o lista de episodios del escenario u otro componente * Marcar en caso de observaciones

TABLA 2 **Forma VII.------**

NºSec.

CONTROL DE SUPERPOSICIÓN DE ESCENARIOS

ESCEN**i**[@]	ESCEN**j**[@]	ACTOR		RECURSO		CONTEXTO		**Iij**[*]	OBJETIVO[+]	EPISODIO[+]	OVERLAP[%]
		$A_{\cup ij}$[&]	$A_{\cap ij}$[+]	$R_{\cup ij}$[&]	$R_{\cap ij}$[+]	$C_{\cup ij}$[&]	$C_{\cap ij}$[+]				

[@] Número de escenario [+] Total de coincidencias [&] Unión de ítems [*] $I_{ij} = (\alpha * C_{\cap ij} + \beta * A_{\cap ij} + \gamma * R_{\cap ij}) / (\alpha * C_{\cup ij} + \beta * A_{\cup ij} + \gamma * R_{\cup ij})$

[%] Marcar en caso de overlap

TABLA 2 **Forma VIII.------**
NºSec.

CONTROL DE OCURRENCIA DE ACTORES

ACTOR*	ESCENARIOS°	EPISODIOS°	LEL+

° Totales * Sin incluir los usos duplicados + Símbolo del LEL (Sí/No)

TABLA 2 **Forma IX.--------**

Nº Sec.

CONTROL DE USO DE RECURSOS

RECURSO*	ESCENARIOS°	EPISODIOS°	LEL+

° Totales * Sin incluir los usos duplicados + Símbolo del LEL (Sí/No)

TABLA 2 **Forma X.--------**
NºSec.

SÍMBOLOS DEL LEL NO UTILIZADOS

SÍMBOLO	LEL[+]	IMPACTOS[º]	REF.[*]

º Totales
+ Sujeto, Objeto, Verbo, Estado, Otra Clase
* Sin incluir los usos duplicados

TABLA 2 **Forma XI.--------**

NºSec.

CONTROL DE IMPACTOS VERSUS ESCENARIOS / EPISODIOS

SUJETO	IMPACTO[+]	ESCENARIOS / EPISODIOS [%]	ACTORES	OBS[*]

[+]Número de Impacto [%]Lista de Escenarios / Episodios [%]Marcar en caso de diferencias

TABLA 2 **Forma XII--------**
NºSec.

LISTA CANDIDATA DE DEOs
OBSERVACIONES

Correcciones propuestas			
Nro.Forma	Nro.Sec.*	Ítem	Observación

Información a recabar			
Nro.Forma	Nro.Sec.*	Ítem	Observación

* Si corresponde

TABLA 2 **Forma XIII--------**

NºSec.

COMENTARIO GENERAL

F.4. Gestión de Inspección

F.4.1. Guías de Instrucción de Gestión de Inspección

Tabla 3. **REPORTE DE RESULTADOS DE INSPECCIÓN.**
***Objetivo**:* Evaluar la calidad del proceso de inspección y de los inspectores, con el fin de mejorar el proceso.
***Proceso General**:* • Completar los formularios según el siguiente orden: ♦ Formulario I ♦ Para cada uno de los Tipos de Evaluación de DEOs: ◊ Formulario III ◊ Formulario IV ♦ Formulario V ♦ Formulario II • En el caso de los Formularios III y IV, utilizar una hoja por cada Tipo de Evaluación de la severidad de los DEOs, numerándolas correlativamente. • Una vez que se han procesado todos los formularios, archivar los mismos siguiendo estrictamente los números de formulario. • Una vez finalizada la verificación, volcar el resumen estadístico en el Formulario II en base a Formularios III a V, y realizar un comentario general acerca de la calidad de la inspección, sobre la calidad del material inspeccionado y sobre todo otro aspecto que corresponda. • En el Formulario I se registra la disposición final del material inspeccionado. Por ejemplo: aprobado, aprobado con correcciones menores, re-inspeccionar.
***Análisis**:* Los formularios III y IV permiten evaluar cuantos defectos son detectados por el método de inspección o encontrados espontáneamente por el inspector sin uso de la heurística. También estos formularios permiten detectar la severidad de los defectos hallados. El formulario V permite evaluar errores en la inspección, ya sea provenientes del método seguido o del inspector, es decir, determinar si el método o el inspector detecta DEOs que son rechazados como tales.

Forma I. **CARÁTULA**.
***Objetivo**:* Identificar el proyecto y el conjunto de escenarios a consistir.
***Pasos**:* • Registrar la información básica y disponible del proyecto y de los escenarios. • Completar el Nro. de Tabla a inspeccionar: Tabla 1 (Intra Inspección) o Tabla 2 (Inter Inspección) • Al finalizar el proceso, registrar el esfuerzo de la reunión y la disposición final de los escenarios.
***Análisis**:* No corresponde.

Forma II. **RESUMEN DE DEOs**.
***Objetivo**:* Brindar una idea general de los defectos encontrados y de la calidad del proceso de inspección.
***Pasos**:* • Completar el Nro. de Tabla en base al formulario I. • Completar la primera tabla del formulario contabilizando la cantidad de defectos en base al formulario III. • Completar la segunda tabla del formulario contabilizando la cantidad de defectos en base a las hojas del formulario IV. • Completar la tercera tabla del formulario contabilizando la cantidad de defectos en base a las hojas del formulario IV. • Completar la cuarta tabla del formulario contabilizando la cantidad de defectos en base al formulario V. • Registrar en la celda **Observaciones** la calidad de los escenarios inspeccionados y la calidad del proceso de inspección.
***Análisis**:* Si hay errores del método, éstos deberán ser corregidos posteriormente. Si hay errores del inspector, éste deberá ser instruido sobre la forma de aplicar el método. Si hay una cantidad excesiva de defectos Fundamentales (++), se deberá instruir a los autores de los escenarios sobre el proceso de construcción de los mismos.

Forma III. **EVALUACIÓN DEL PROCESO DE INSPECCIÓN**.
***Objetivo**:* Evaluar el diseño del proceso de inspección.
***Pasos**:* • Generar una hoja por cada tipo de severidad de los defectos (++, +- y --) y numerarlas correlativamente. • Completar en cada hoja el Nro. de Tabla en base al formulario I. • En base a Lista Candidata de DEOs, completar con los números de escenarios con defectos, distinguiendo si son Correcciones a realizar por los autores o Dudas a recabar en el Universo de Discurso, según el nro. de forma donde fue detectado el defecto y su grado de severidad (ver Anexo). También se clasifican por el mecanismo de detección, si correspondió al método o fueron encontrados en forma espontánea por el inspector.
***Análisis**:* No corresponde

Forma IV. **DEOs POR COMPONENTE**.
***Objetivo**:* Evaluar los defectos encontrados.

***Pasos**:*
- Generar una hoja por cada tipo de severidad de los defectos (++, +- y --) y numerarlas correlativamente.
- Completar en cada hoja el Nro. de Tabla en base al formulario I.
- En base a Lista Candidata de DEOs, completar con los números de escenarios con defectos, distinguiendo si son detectados por el método o espontáneo por el inspector, según el componente del escenario donde el defecto fue detectado y su grado de severidad (ver Anexo).

***Análisis**:*
No corresponde

Forma V. **DEOs RECHAZADOS POR FORMA**.

***Objetivo**:* Evaluar los defectos rechazados por fuente de generación: el proceso o el inspector.

***Pasos**:*
- Completar el Nro. de Tabla en base al formulario I.
- En base a Lista Candidata de DEOs, completar con los números de escenarios con defectos detectados en la Preparación pero descartados durante la Reunión, distinguiendo si el error fue provocado por el método o por el inspector, y según la forma donde se detectó.

***Análisis**:*
No corresponde

Anexo. **TIPO DE EVALUACIÓN DE DEOs POR SEVERIDAD**.

***DEOs Fundamentales (++)**:* Son detecciones muy importantes. Corresponde a todo defecto que altera el significado del escenario.

***Ejemplos**:*
- Tabla 1
  - Objetivo no coincide con el Título.
- Tabla 2
  - Inexistencia física de un Sub-Escenario.
  - Omisión de información (impactos del LEL no cubiertos).
  - Discrepancias entre los Sujetos de los impactos del LEL y los Actores de los Episodios.
  - Overlap de escenarios.

***DEOs Organizacionales (+-)**:* Son detecciones medianamente importantes. Corresponde a todo defecto que presenta inconsistente al escenario sin alterar el significado del mismo.

***Ejemplos**:*
- Tabla 1
  - Agregar o Eliminar actores o recursos del componente Actores o del componente Recursos respectivamente.
  - Separar un episodio en dos episodios.
  - Dudas sobre información faltante o confusa en algún componente del escenario.
  - Mal uso de un símbolo del LEL.

° Errores de sintaxis. • Tabla 2 ° Un símbolo del LEL no utilizado en ningún escenario.
***DEOs de Presentación (–)**:* Son detecciones poco importantes. Corresponde a todo defecto cuya corrección es prácticamente aclaratoria, permite completar y mejorar la descripción del escenario sin alterar su significado.
***Ejemplos**:* • Tabla 1 ° Símbolos del LEL no detectados (marcados) en el escenario ° Verbos mal conjugados en el tiempo, principalmente en episodios. ° Falta la palabra DEBE en una restricción. ° Símbolos mal identificados que no pertenecen al LEL. ° Errores de sintaxis.

F.4.2. Formularios de Gestión de Inspección

TABLA 3 **Forma I**

Reporte de Resultados de Inspección

Escenarios inspeccionados

Proyecto: .....

Grupo/Autor(es): .....

Versión: ..... Fecha: .....

Cant. de Escenarios: ..... Cant. de Episodios: .....

Datos Generales de la Inspección

Tabla Nro: ..... Tipo de inspección: .....

Preparación:

Fecha entrega material: ..... Esfuerzo: .....

Cant. de Páginas llenas: ..... Cant. de líneas llenas: .....

Reunión:

Fecha: ..... Esfuerzo: .....

Inspector: .....

Moderador: .....

Escriba: .....

Productores: .....

Disposición final: .....

TABLA 3 **Forma II**

Resumen de DEOs – Tabla Nro.____

Tipo de detección	++	+-	--	Total
Correcciones espontáneas				
Correcciones debido al método				
Dudas espontáneas				
Dudas del método				
Total				

Tipo de DEO's	++	+-	--	Total
Discrepancias				
Errores				
Omisiones				
Total				

Componentes	Discrepancias	Errores	Omisiones	Total
Título				
Objetivo				
Contexto				
Recursos				
Actores				
Episodios				
Excepciones				
Restricciones				
No Especifico				
Total				

Errores	Total
del Método	
del Inspector	
Total	

++ DEO que altera el significado del escenario
+- DEO que altera estructuralmente el escenario sin modificar su significado
-- DEO que marca una mínima incompletitud del

Observaciones:

……
……
……
……
……
……
……

TABLA 3 **Forma III._____**

Nº Sec.

EVALUACIÓN DEL PROCESO DE INSPECCIÓN - TABLA Nro. ____

Tipo de Evaluación [@]

++	+-	--

	CORRECCIONES*		***DUDAS*** [$]	
Forma	**ESPONTANEOS** [^]	**DEL METODO** [!^]	**ESPONTANEOS** [^]	**DEL METODO** [!^]
II				
III				
IV				
V				
VI				
VII				
VIII				
IX				
X				
XI				
XII				
XIII				

[@] ++DEO que altera el significado del escenario +-DEO que altera estructuralmente el escenario sin modificar su significado --DEO que marca una mínima incompletitud del escenario

***** DEO cuya corrección no requiere de la intervención del autor del escenario o la elicitación en el UdeD.

[$] DEO cuya corrección requiere la interpretación, validación y/o elicitación con el autor del escenario o en el UdeD.

[^] DEO que se detecta espontáneamente al observar el escenario [i^] DEO que se detecta a través del método

TABLA 3 **Forma IV._____**

Nº Sec.

DEO's POR COMPONENTE – TABLA Nro. ____

Tipo de Evaluación[@]

++	+-	--

	DISCREPANCIAS		***ERRORES***		***OMISIONES***	
Componente	**ESPONTANEO**[^]	**DEL MÉTODO**[!^]	**ESPONTANEO**[^]	**DEL MÉTODO**[!^]	**ESPONTANEO**[^]	**DEL MÉTODO**[!^]
Título						
Objetivo						
Contexto						
Recursos						
Actores						
Episodios						
Excepciones						
Restricciones						
No Específico						

[@] ++DEO que altera el significado del escenario +-DEO que altera la estructura del escenario sin modificar su significado --DEO que marca una mínima incompletitud del escenario

[^]DEO que se detecta espontáneamente al observar el escenario [!^]DEO que se detecta a través del método

TABLA 3 *Forma V*

DEOs RECHAZADOS POR FORMA - TABLA Nro. ____

	ERRORES DE INSPECCION	
Forma	***provenientes del METODO***	***provenientes del INSPECTOR***
II		
III		
IV		
V		
VI		
VII		
VIII		
IX		
X		
XI		
XII		
XIII		

Apéndice G
Relación entre Escenario y Caso de Uso

**«Software is the only engineering discipline
in which the equivalent of changing the wing on an airplane
constitutes maintenance »**

Proverbio industrial, citado por Jim Highsmith

G.1. Relación entre los tipos de escenarios y de casos de uso

Como ya se ha mencionado en la sub-sección 2.8.1, para algunos autores escenarios y casos de uso son sinónimos, mientras que otros hacen una distinción importante; todo depende de la definición que se esté utilizando de ambos modelos. En esta tesis se está considerando una definición amplia de casos de uso, que permite establecer una relación estructural entre el modelo de escenario y el modelo de caso de uso.

Caso de Uso: <Nombre del caso de uso>

Alcance: <Sistema bajo estudio>

Nivel: <Grado de detalle del caso de uso>

Actor Principal: <Nombre del actor que inicia el caso de uso>

Involucrados e intereses: <Objetivos e intereses de los involucrados>

Precondición: <Condición que debe cumplirse para iniciar el caso de uso>

Disparador: <Evento que inicia el caso de uso>

Garantías Mínimas: <Afirmaciones de lo que será verdad al finalizar el caso de uso por éxito o por fracaso>

Garantías de Éxito: <Afirmaciones de lo que será verdad al finalizar el caso de uso por éxito>

Escenario Principal de éxito:
1. <Pasos del Escenario Principal>
2. ...
3. ...

Extensiones:
3a. <Condición de la Extensión>
    3a1. <Pasos de la Condición>
    3a2. ...

Figura G-1. Modelo de Caso de Uso

SDRES puede adaptarse utilizando el modelo de caso de uso propuesto por [Cockburn 00] (ver Figura G-1) en lugar del modelo de escenario (ver Figura 5-3), para ello deben establecerse las relaciones existentes entre ellos y sus componentes.

Tanto los escenarios actuales como los escenarios futuros se reemplazan por casos de uso del negocio. Los casos de uso que se proponen en [Cockburn 00] son de tres tipos según su alcance:

i) Caso de uso del negocio, que describe las actividades del negocio. Se estudia la organización en sí misma.
ii) Caso de uso del sistema, que lo describe externamente, como caja negra. El actor principal es un usuario frente a la computadora.
iii) Caso de uso de diseño, que describe al sistema internamente, como caja blanca. El actor principal es un objeto usuario.

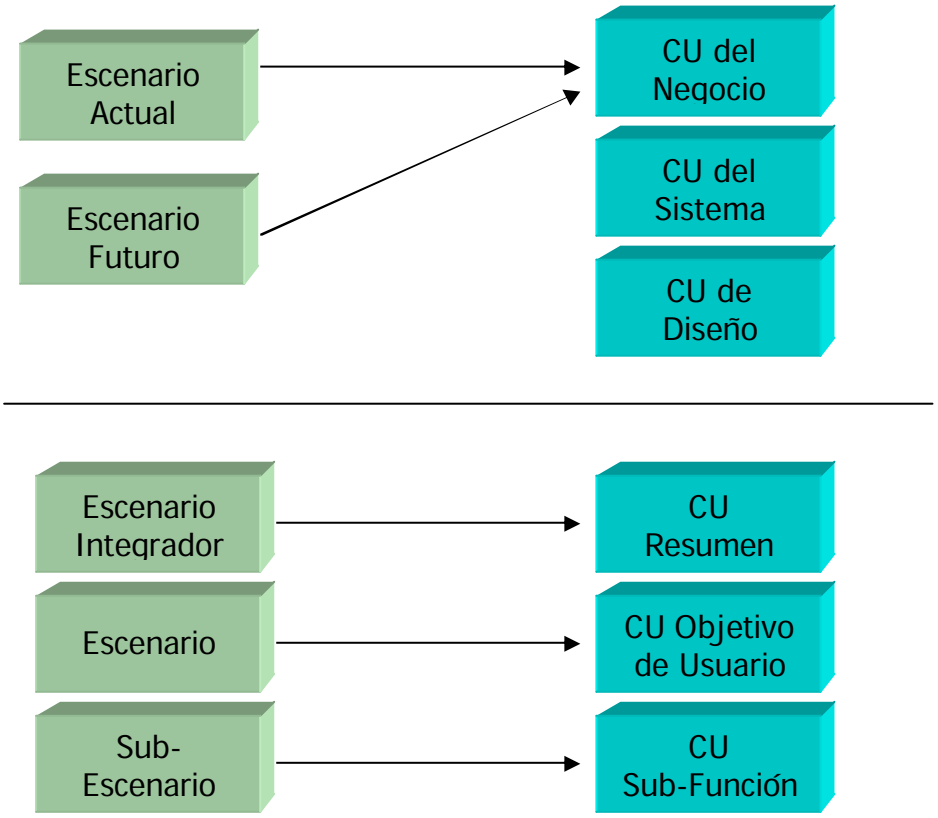


Figura G-2. Relaciones entre tipos de escenarios y tipos de casos de uso[180]

Por otro lado, los casos de uso se distinguen por su nivel de detalle en:

i) Nivel Resumen, que describen múltiples sesiones de casos de uso de objetivo de usuario.
ii) Nivel Objetivo de Usuario, que describen un objetivo particular e

[180] CU: Caso de Uso

inmediato de un actor principal.

iii) Nivel Sub-función, que describe objetivos parciales de un caso de uso de objetivo de usuario o de otra sub-función.

Los escenarios se representan por casos de uso de nivel objetivo de usuario mientras que los sub-escenarios se representan con casos de uso de nivel sub-función. Los escenarios integradores se representan con casos de uso de nivel resumen, pues todos sus pasos pueden ser casos de uso de objetivo de usuario o casos de uso resumen. La Figura G-2 muestra las relaciones mencionadas respecto al alcance y al nivel de detalle entre los escenarios y los casos de uso.

Además debe considerarse que un caso de uso alberga todos los posibles casos para la obtención del objetivo o su fracaso. Mientras que una situación con variantes puede describirse en uno o varios escenarios, dependiendo de la complejidad de cada variante o caso alternativo. Cuando se describen en un único escenario, se encontrarán en él episodios condicionales según las variantes. En los casos de uso los pasos nunca llevan una condición en la variante normal (denominada Escenario Principal de Éxito), el resto de las variantes y excepciones se describen en el componente Extensiones, donde sí se aplica una condición a un conjunto de pasos que representan una determinada variante o excepción (denominados Escenarios Secundarios).

G.2. Relación entre los componentes de escenarios y de casos de uso

El caso de uso no tiene un componente Objetivo como el escenario, sino que el Nombre del caso de uso (correspondiente al Título del escenario) representa el objetivo del actor principal. En el caso del escenario se describe el objetivo de la situación, no el de alguien en particular. Los actores participantes en un escenario se identifican en el caso de uso dentro de los componentes Actor Principal e Involucrados, pero en este último componente pueden identificarse actores que no participan en el caso de uso pero que tienen algún interés en su realización. Los recursos identificados en un escenario no tienen un parangón en el caso de uso.

El contexto del escenario incluye tres sub-componentes: ubicación geográfica, ubicación temporal y precondición. En el caso de uso sólo se considera la precondición. En el componente alcance, siendo que se describe el negocio, puede identificarse la organización o departamento o área de la empresa bajo estudio en dicho caso de uso, dado lo cual puede haber algún tipo de relación con la ubicación geográfica.

En un escenario no se indica si se trata de un escenario integrador, un escenario o un sub-escenario. En un caso de uso se explicita en el componente Nivel el grado de detalle que presentan sus pasos.

El componente Disparador del caso de uso puede estar contenido en la Precondición del escenario o ser el primer episodio del mismo. En algunos

casos el Disparador indica un evento temporal, con lo cual puede relacionarse con la Ubicación Temporal. Los componentes del caso de uso Garantías Mínimas y Garantías de Éxito, representando respectivamente post-condición por éxito o fracaso y post-condición por éxito no se manifiestan en el modelo de escenario.

El componente Escenario Principal de Éxito del caso de uso incluye todos los pasos que representan el flujo normal de eventos, y en el componente Extensiones se incluyen los pasos que representan flujos alternativos ya sea de éxito o de fracaso del caso de uso. En el escenario, los flujos de eventos normales y de eventos alternativos se representan en el componente Episodios, y la interrupción de episodios se representa en el componente Excepciones. En ambos modelos, los pasos o episodios muestran el orden secuencial en que se realizan, y también admiten indicar un orden paralelo. Un paso o episodio puede ser a su vez el nombre de un caso de uso o de escenario respectivamente.

Otros componentes opcionales en el modelo de caso de uso son:

- ⇨ Lista de Variaciones Tecnológicas y de Datos: distintas formas de realizar un paso o con distintos tipos de datos.
- ⇨ Contexto de Uso: breve descripción del caso de uso.
- ⇨ Requisitos Especiales[181]: lista de RNF vinculados al caso de uso.
- ⇨ Temas Pendientes: lista de preguntas o dudas sobre el caso de uso.

Caso de Uso / **Escenario**	Nombre	Alcance	Actor Principal	Involucrados e Intereses	Precondición	Disparador	Garantías Mínimas	Garantías de Éxito	Esc.Principal de Éxito	Extensiones	Lista de Variaciones	Requisitos Especiales
Título	v v											
Objetivo	v											
Ubic.Geográfica		v										
Ubic. Temporal						v						
Precondición					v v	v v						
Recursos												
Actores			v v	v								
Episodios									v v	v v	v	
Restricción												v v
Excepciones										v v		

Tabla G-1. Relación entre componentes de Escenario y de Caso de Uso

En el modelo de escenario, los Requisitos Especiales se incluyen en el

[181] Este componente está disponible en la plantilla de caso de uso de RUP.

atributo Restricción de cada episodio, y los Temas Pendientes se describen en el componente temporal Dudas, el cual es eliminado luego de que el escenario es verificado y validado. El componente Contexto de Uso no tiene cotejo en el modelo de escenario, donde realmente es innecesario. El componente Lista de Variaciones Tecnológicas y de Datos representa información que en el modelo de escenario se incluye como parte de la descripción del episodio donde se corresponde. Otras veces esta información puede estar registrada en algún símbolo del LEL tipo Objeto correspondiente a un Recurso del escenario o en algún símbolo del LEL tipo Verbo incluido en el episodio.

En la Tabla G-1 se muestra una matriz cuyas filas representan los componentes del escenario y sus columnas los componentes del caso de uso. La intersección de una fila – columna marcada con **v v** indica la existencia de una correlación fuerte entre ambos componentes, mientras que la marca **v** indica una posible relación. La fila pintada de gris así como las dos columnas pintadas de gris indican que no hay ningún tipo de relación entre ambos modelos para dichos componentes.

Referencias

[Abrahamsson 02] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., "Agile software development methods: Review and Analysis", VTT Publications 478, Finlandia, 2002, http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf accedida el 05-04-2005.

[Ackerman 89] Ackerman, A.F., Buchwald, L.S., Lewsky, F.H., "Software Inspections: An Effective Verification Process", IEEE Software, Vol.6, Nº3, 1989, pp. 31-36.

[AgileAlliance 01] "Agile Alliance", Utah, 11-13 de Febrero 2001, http://www.agilealliance.org/, accedida el 20-02-2005.

[AgileManifesto 01] "Manifesto for Agile Software Development", Utah, 13 de Febrero 2001, http://www.agilemanifesto.org/, accedida el 20-02-2005.

[Ahern 03] Ahern, D.M., Clouse, A., Turner, R., "CMMI Distilled: A Practical Introduction to Integrated Process Improvement", Addison-Wesley, 2º edición, 2003.

[Alexander 01] Alexander, I., "Being Clear: Requirements are EITHER Needs OR Specifications", columna en la revista electrónica Requirenautics Quarterly, BCS Requirements Engineering Specialist Group, Junio 2001.

[Alexander 04] Alexander, I., Maiden, N. (eds.), "Scenarios, Stories, Use Cases. Through the Systems Development Life-Cycle", 1º edición, John Wiley & Sons, Septiembre 2004.

[Alexander 04b] Alexander, I., "A Taxonomy of Stakeholders: Human Roles in System Development", International Journal of Technology and Human Interaction, Vol. 1,Nº 1, 2005, pp.23-59.

[Alexander 05] Alexander, I., Robertson, S., "Understanding Project Sociology by Modeling Stakeholders", IEEE Software, Vol. 21, Nº1, Enero/Febrero, 2004, pp.23-27.

[Alford 79] Alford, M., Lawson, J., "Software Requirements Engineering Methodology (Development)", RADC-TR-79-168, US Air Force Rome Air Development Center, Junio 1979.

[Almela 05] Almela, R., Cantos, P., Sánchez, A., Sarmiento, R., Almela, M., "Frecuencias del español. Diccionario y estudios léxicos y morfológicos", Madrid: Universitas, ISBN: 84-7991-171-9, 591 páginas, 2005.

[Alspaugh 99] Alspaugh, T.A., Antón, A.I., Barnes, T., Mott, B.W., "An Integrated Scenario Management Strategy", International Symposium On Requirements Engineering (RE´99), Limerick, Irlanda, IEEE Computer Society Press, 1999, pp.142-149.

[Andrews 91] Andrews, D.C., "JAD: A crucial dimension for rapid applications development", Journal of Systems Management, Vol.42, Nº3, Marzo 1991, pp.23-31.

[Arango 88] Arango, G., "Domain engineering for software reuse", Ph.D. thesis, Department of Computer Science, University of California, Irvine, 1988.

[Arango 89] Arango, G., Prieto-Diaz, R., "Domain analysis: Concepts and research directions", en Domain Analysis: Acquisition of Reusable Information for Software Construction, editores R. Prieto-Diaz y G. Arango, IEEE Computer Society Press, Mayo 1989.

[Babich 86] Babich, W.A., "Software Configuration Management", Addison-Wesley, Reading, Massachusetts, 1986.

[Balzer 83] Balzer, R., Cheatham, T.E., Green, C., "Software technology in the 1990s: using a new paradigm", IEEE Computer, Noviembre 1983.

[Basili 81] Basili, V.R., Weiss, D., "Evaluation of a Software Requirements Document by Analysis of Change Data, Fifth International Conference on Software Engineering, Los Alamitos, CA, IEEE Computer Society Press, 1981, pp.314-323.

[Basili 96] Basili, V., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sorumgard, S., Zelkowitz, M., "The Empirical Investigation of Perspective-based reading", Journal of Empirical Software Engineering, Vol.2, Nº1, 1996, pp.133-164.

[Beck 99] Beck, K., "Embracing change with Extreme Programming", IEEE Computer, Vol. 32, Nº10, Octubre 1999.

[Beck 00] Beck, K., "Extreme Programming Explained: Embrace Change", Addison-Wesley, 2000.

[Beedle 99] Beedle, M., et al., "SCRUM: An Extension Pattern Language for Hyperproductive Software Development", en Pattern Languages of Program Design, Vol.4, 1999, pp.637-651.

[Belady 76] Belady, L.A., Lehman, M.M., "A Model of Large Program Development", IBM Systems Journal, Vol.15, Nº3, 1976, pp. 225-252.

[Bell 76] Bell, T.E., Thayer, T. A., "Software Requirements: are they really a problem?", Second International Conference on Software Engineering, 1976, pp.61-68.

[Ben Achour 98] Ben Achour, C., Rolland, C., Souveyet, C., "A Proposal for Improving the Quality of Scenario Collections", anales de 4th International Workshop on RE: Foundations of Software Quality, REFSQ'98, Presses University of Namur (http://sunsite.informatik.rwth-aachen.de/CREWS), 1998, pp. 29-42.

[Ben Achour 99] Ben Achour, C., Rolland, C., Maiden, N.A.M., Souveyet, C., "Guiding Use Case Authoring: Results of an Empirical Study", International Symposium On Requirements Engineering (RE´99), Limerick, Irlanda, IEEE Computer Society Press, 1999, pp.36-43.

[Berry 02] Berry, D., "The Inevitable Pain of Software Development, Including of Extreme Programming, Caused by Requirements Volatility", International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Essen, Alemania, 2002, http://www-di.inf.puc-rio.br/~julio/tcre-site/p2.pdf, accedido el 28-03-2005.

[Bertolin 98] Bertolin, J.C.G., "Uma Análise da Aplicação de Técnicas da Psicologia para a Elicitação de Requisitos de Software", tesis de maestría, Universidade Federal do Rio Grande do Sul, Instituto de Informática, Brasil, 1998.

[Boehm 75] Boehm, B.W., et al., "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software", IEEE TSE, Vol.1, Nº1, Marzo 1975, pp.125-133.

[Boehm 76] Boehm, B.W., "Software Engineering", IEEE Transactions

Computers, Vol.25, Nº12, Diciembre 1976, pp.1226-1241.

[Boehm 81] Boehm, B.W., "Software Engineering Economics", Englewood Cliffs, NJ: Prentice-Hall, 1981.

[Boehm 84] Boehm, B.W., "Verifying and Validating Software Requirements and Design Specifications", IEEE Software, Vol.1, Nº1, Enero 1984, pp.75-88. Reimpreso en el libro System and Software Requirements Engineering, editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, Los Alamitos, CA, 1990.

[Boehm 88] Boehm, B.W., "A Spiral Model of Software Development and Enhancement", IEEE Computer, Vol.21, Nº5, Mayo 1988, pp.61-72.

[Boehm 96] Boehm, B.W., In, H., "Identifying Quality-Requirement Conflicts", IEEE Software, Vol.13, Nº2, 1996, pp.25-35.

[Boehm 02] Boehm, B., "Get Ready for Agile Methods, with Care", IEEE Computer, Vol.35, Nº1, Enero 2002, pp.64-69.

[Booch 91] Booch, G., "Object Oriented Design with Applications", The Benjamin Cumming Publishing Company, Redwood City, CA, 1991.

[Booch 92] Booch, G., "Object-Oriented Analysis and Design", The Benjamin Cummings Publishing Company, Redwood City, CA, 1992.

[Booch 94] Booch, G., "Scenarios", Report on Object Analysis and Design, Vol.1, N°3, 1994, pp.3-6.

[Booch 99] Booch, G., Rumbaugh, J., Jacobson, I., "The Unified Modeling Language User Guide", Addison-Wesley, Reading, MA, 1999.

[Breitman 98] Breitman, K.K, Leite, J.C.S.P., "A Framework for Scenario Evolution", IEEE International Conference on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp.214-221.

[Breitman 99] Breitman, K.K., Leite, J.C.S.P., Finkelstein, A., "The World´s a Stage: A Survey on Requirements Engineering using a Real-Life Case Study", Journal of the Brazilian Computer Society, Sociedade Brasileira de Computação, Vol.6, Nº1, 1999, pp.13-37.

[Breitman 01] Breitman, K.K., Leite, J.C.S.P., "Requirements Elicitation through Scenarios", 5th IEEE International Symposium on Requirements Engineering (RE'01), tutorial, Toronto, Canadá, Agosto 2001.

[Breitman 03] Breitman, K.K., Leite, J.C.S.P., "Ontology as a requirements engineering product", 11th IEEE International Conference on Requirements Engineering (RE 2003), Monterey Bay, CA, USA, IEEE Computer Society 2003, ISBN 0-7695-1980-6, Septiembre 2003, pp.309-319.

[Breitman 05] Breitman, K.K., Leite, J.C.S.P., Berry, D.M., "Supporting scenario evolution", Requirements Engineering Journal, Springer-Verlag London Ltd, Vol.10, Nº2, Mayo 2005, pp.112-131.

[Brooks 87] Brooks, F.P., "No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Computer Society Press, Vol.20, Nº4, Abril 1987, pp.10-19.

[Brooks 87b] Brooks, F.P., chairman "Report of the Defense Science Board Task Force on Military Software", Office of the Under Secretary of Defense for Acquisition, U.S. Department of Defense, Washington, DC, Septiembre 1987.

[Brown 92] Brown, A.W., Earl, A.N., McDermid, J.A., "Software Engineering Environments: Automated Support for Software Engineering", McGraw-Hill, Londres, 1992.

[Bubenko 80] Bubenko, J., "Information Modeling in the context of system development", IFIP Congress'80, 1980, pp.395-411.

[Bubenko 93] Bubenko, J.A., Wrangler, B., "Objectives driven capture of business rules and information systems requirements", IEEE Conference on Systems, Man and Cybernetics, 1993.

[Burstin 84] Burstin, M., "Requirements Analysis of Large Software Systems", Ph.D. dissertation, Tel-Aviv University, Israel, 1984.

[Carroll 95] Carroll, J., "Introduction: The Scenario Perspective on System Development", en el libro Scenario-Based Design: Envisioning Work and Technology in System Development, editor J. Carroll, John Wiley & Sons, Nueva York, 1995, pp.1-18.

[Carroll 95b] Carroll, J.M., Rosson, M.B., "Narrowing the Specification Implementation Gap in Scenario-based Design", en el libro Scenario-Based Design: Envisioning Work and Technology in System Development, editor J. Carroll, John Wiley & Sons, Nueva York, 1995, pp.247-278.

[Carter 01] Carter, R.A., Antón, A.I., Dagnino, A., Williams, L., "Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model", Fifth IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp.94-101.

[Castro 02] Castro, J., Kolp, M., Mylopoulos, J., "Towards Requirements-Driven Information Systems Engineering: The Tropos Project", Information Systems Journal, Elsevier, Amsterdam, Vol.27, 2002, pp.365-389.

[CHAOS 95] CHAOS Report, Standish Group, Enero 1995, http://www.standishgroup.com/

[Chen 76] Chen, P., "The entity-relationship model: Towards a unified view of data", ACM Transactions on Database Systems, Vol.1, Nº1, 1976.

[Cheng 96] Cheng, B., Jeffrey, R., "Comparing Inspection Strategies for Software Requirements Specifications", 1996 Australian Software Engineering Conference, 1996, pp.203-211.

[Chrissis 03] Chrissis, M.B., Konrad, M., Shrum, S., "CMMI: Guidelines for Process Integration and Product Improvement", Addison-Wesley, 2003.

[Chung 00] Chung, L., Nixon, B.A., Yu, E., "Non-Functional Requirements in Software Engineering", Kluwer Academic Publishers, 2000.

[Ciolkowski 97] Ciolkowski, M., Differding, C., Laitenberger, O., Münch, J., "Empirical Investigation of Perspective-based reading: A Replicated Experiment", ISERN Report Nº 97-13, 1997.

[CMM 95] Software Engineering Institute, "The Capability Maturity Model: Guidelines for Improving the Software Process", Carnegie Mellon University, Addison Wesley, 1995, http://www.sei.cmu.edu/cmm/cmm.html/, accedida el 14-02-2005.

[CMMI 06] Software Engineering Institute, "Capability Maturity Model Integration", CMMI-DEV v1.2, CMU/SEI-2006-TR-008, Carnegie Mellon University, 2006, http://www.sei.cmu.edu/cmmi/, accedida el 14-10-2007.

[Coad 91] Coad, P., Yourdon, E., "Object-Oriented Analysis", 2º edición, Yourdon Press, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[Coad 99] Coad, P., et al., "Feature-Driven Development", en Java Modeling in Color with UML, Prentice Hall, 1999.

[Cockburn 95] Cockburn, A., "Structuring Use Cases with Goals", Reporte Técnico, Humans and Technology, Salt Lake City, UT, 1995,

http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm, accedida el 17-03-2006.

[Cockburn 00] Cockburn, A., "Writing Effective Use Cases", Humans and Technology, Addison-Wesley, 2000.

[Cockburn 02] Cockburn, A., "Agile Software Development", Addison-Wesley, 2002.

[Codd 79] Codd, E., "Extending the database relational model to capture more meaning", ACM Transactions on Database Systems, Vol.4, 1979.

[Codd 82] Codd, E., "Relational databases: A practical foundation for productivity", Communications of the ACM, 1982.

[Coleman 94] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes, P., "Object-Oriented Development: The Fusion Method", Prentice-Hall, Englewood Cliffs, NJ, 1994.

[Conradi 98] Conradi, R., Westfechtel, B., "Version Models for Software Configuration Management", ACM Computing Surveys, Vol.30, 1998, pp.232-282.

[Constantine 98] Constantine, L., "Joint Essential Modelling, User Requirements Modelling for Usability", International Conference on Requirements Engineering (Tutorial Notes), Colorado Springs, Constantine & Lockwood, 1998.

[Coulin 05] Coulin, C., Zowghi, D., "Requirements Elicitation for Complex Systems: Theroy and Practice", en el libro "Requirements Engineering for Sociotechnical Systems", capítulo III, Information Science Publishing, Maté & Silva editores, Londres, 2005, pp.37-52.

[Craciunescu 04] Craciunescu, O., Gerding-Salas, C., Stringer-O'Keeffe, S., "Machine Translation and Computer-Assisted Translation: a New Way of Translating?", Translation Journal (http://accurapid.com/journal), ISSN 1536-7207, Vol.8, Nº3, Julio 2004.

[Crnkovic 99] Crnkovic, I., Funk, P., Larsson, M., "Processing Requirements by Software Configuration Management", 25th Euromicro Conference (EUROMICRO'99), IEEE Computer Society, Milán, Italia, Vol.2, Septiembre 1999, pp.2260-2265.

[Cruz Neto 04] Cruz Neto, G., Gomes, A.S., Castro, J.B., "Mapeando Diagramas da Teoria da Atividade em Modelos Organizacionais Baseados em i*", VII Workshop on Requirements Engineering (WER'04), Tandil, Buenos Aires, Diciembre 2004, pp.39-50.

[CUP 03] "ConsultIT's Framework: CUP – ConsultIT Unified Process", ConsultIT Competence centers (http://www.consultit.no/consultit/), Noruega, 2003, https://intranett.consultit.no/rammeweb/ (ConsultIT user account required), accedida el 25-02-2005.

[Cysneiros 99] Cysneiros, L.M., Leite, J.C.S.P., "Integrating Non-Functional Requirements into Data Modeling", IEEE International Symposium on Requirements Engineering, RE'99, Limerick, Irlanda, 1999, pp.162-171.

[Cysneiros 01] Cysneiros, L.M., Leite, J.C.S.P., "Driving Non-Functional Requirements to Use Cases and Scenarios", anales del XV Simpósio Brasileiro de Engenharia de Software SBES 2001, Rio de Janeiro, 2001, pp. 7-20.

[Cysneiros 04] Cysneiros, L.M., Yu, E., "Non-Functional Requirements Elicitation", en el libro "Perspectives on Software Requirements", Kluwer Academic Publishers, Estados Unidos, capítulo 6, 2004, pp.115-138.

[Daly 77] Daly, E., "Management of Software Development", IEEE Transactions Computers, Vol.3, Nº3, Mayo 1977, pp.229-242.

[Dano 97] Dano, B., Briand, H., Barbier, F., "An Approach Based on the Concept of Use Case to Produce Dynamic Object-Oriented Specifications", Third IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, California, 1997, pp.54-64.

[Dardenne 93] Dardenne, A., van Lamsweerde, A., Fickas, S., "Goal-directed requirements acquisition", Science of Computer Programming, Vol.20, 1993, pp. 3-50.

[Davis 88] Davis, A.M., Bersoff, E.H., Comer, E.R., "A Strategy for Comparing Alternative Software Development Life Cycle Models", IEEE TSE, Vol.14, Nº10, Octubre 1988, pp.1453-1461. Reimpreso en "Software Requirements Engineering", editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.408-415.

[Davis 93] Davis, A. M., "Software Requirements: Objects, Functions and States", Prentice Hall, Englewood Cliffs, NJ, 2º edición, 1993.

[Davis 93b] Davis, A., et al., "Identifying and Measuring Quality in a Software Requirements Specification", First International Software Metrics Symposium, IEEE Computer Society Press, Baltimore, Mayo 1993, pp.141-152. Reimpreso en "Software Requirements Engineering", editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.164-175.

[Davis 96] Davis, A., comunicación privada a Merlin Dorfman en 1996, mencionada en [Dorfman 97].

[Davis 99] Davis, A., Leffingwell, D., "Making Requirements Management Work For You", Crosstalk, The Journal of Defense Software Engineering, Vol.12, Nº4, Abril 1999.

[De Bortoli 00] De Bortoli, L.Â, Alencar Price, A.M., "O Uso de Workflow para Apoiar a Elicitação de Requisitos", WER'00 – III Workshop de Engenharia de Requisitos, Río de Janeiro, Brasil, Julio 2000, pp.22-37.

[DeGiacomo 97] DeGiacomo, G., Lespérance, Y., Levesque. H., "Reasoning about concurrent execution, prioritized interrupts and exogenous actions in the Situation Calculus", Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, 1997, pp.1221-1226.

[DeMarco 78] DeMarco, T., "Structured Analysis and System Specification", Englewood Cliffs, NJ, Prentice Hall, 1978.

[Desharnais 98] Desharnais, J., Frappier, M., Khedri, R., Mili, A. "Integration of Sequential Scenarios", IEEE Transactions on Software Engineering, Vol. 24, N° 9, 1998, pp. 695-708.

[Díaz 04] Díaz, I., Pastor, O., Moreno, L., Matteo, A., "Una Aproximación Lingüística de Ingeniería de Requisitos para OO-Method", VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software (IDEAS 2004), Arequipa – Perú, Mayo 2004, 12 páginas.

[DOD 5000.2-R] US Department of Defense, "DoD Regulation Guidance 5000.2-R: Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs", Estados Unidos, Abril 2002.

[Doorn 98] Doorn, J., Kaplan, G., Hadad, G., Leite, J.C.S.P., "Inspección de

Escenarios", WER'98 - Workshop de Engenharia de Requisitos, Maringá, Paraná, Brasil, 1998, pp.57-69.

[Doorn 02] Doorn, J.H., Hadad, G.D.S., Kaplan, G.N., "Comprendiendo el Universo de Discurso Futuro", WER'02 - Workshop en Ingeniería de Requisitos, Valencia, España, pp.117-131, Noviembre 2002.

[Doorn 03] Doorn, J.H., Ridao, M., "Completitud de Glosarios: un Estudio Experimental", WER'03 - Workshop en Ingeniería de Requisitos, Piracicaba-SP, Brasil, pp.317-328, Noviembre 2003.

[Dorfman 97] Dorfman, M., "Requirements Engineering", en el libro "Software Requirements Engineering", editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.7-22.

[Dubois 86] Dubois, E., Hagelstein, E., Lahou, E., Ponsaert, F., Rifaut, A., "A Knowledge Representation Language for Requirements Engineering", IEEE, Vol.74, Nº10, 1986, pp.1431-1444.

[Dubois 94] Dubois, E., Du Bois, P., Dubru, F., "Animating Formal Requirements Specifications for Cooperative Information Systems", International Conference on Cooperative information Systems, Toronto, 1994, pp.101-112.

[Dyer 92] Dyer, M., "Verification-based Inspection", 26º Annual Hawaii International Conference on System Sciences, 1992, pp. 418-427.

[Eberlein 02] Eberlein, A., Leite, J.C.S.P., "Agile Requirements Definition: A View from Requirements Engineering", International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Essen, Alemania, 2002, http://www-di.inf.puc-rio.br/~julio/tcre-site/p4.pdf accedido el 28-03-2005.

[Eliot 01] Eliot, C.W., editor, "The Sayings of Confucius" in "The Harvard Classics and Harvard Classics Shelf of Fiction", 1909-1917, Vol.44, Part.1, Bartleby.com, 2001.

[Encarta 04] MSN Encarta Encyclopedia 04, Microsoft Corporation, 2004, http://encarta.msn.com/encyclopedia_761573731/Dictionary.html

[ESA PSS-05-0] European Space Agency Board for Software Standardisation and Control, Software Engineering Standards, DFI. 50, Issue 2, Febrero 1991.

[Esselink 00] Esselink, B., "A Practical Guide to Localization (Language International World Directory)", John Benjamins Publishing Co. editor, ISBN: 1588110060, 2º edición, Septiembre 2000, 490 páginas.

[Estrada 01] Estrada, H., Martínez, A., Pastor, O., Ortiz, J., Ríos, O.A., "Generación Automática de un Esquema Conceptual OO a Partir de un Modelo Conceptual de Flujo de Trabajo", WER'01 – IV Workshop de Engenharia de Requisitos, Buenos Aires, Argentina, Noviembre 2001, pp.223-245.

[Fagan 74] Fagan, M., "Design and Code Inspections and Process Control in the Development of Programs", IBM Corporation, Reporte Técnico TR 21.572, Nueva York, Diciembre 1974.

[Fagan 76] Fagan, M.E., "Design and Code Inspections to reduce Errors in Program Development", IBM Systems Journal, Vol.15, Nº3, 1976, pp.182-211.

[Fagan 86] Fagan, M.E., "Advances in Software Inspections", IEEE Transactions on Software Engineering, Vol.12, Nº 7, 1986, pp. 744-751.

[Fairley 96] Fairley, R.E., Thayer, R.H., "The Concept of Operations: The Bridge

from Operational Requirements to technical Specifications", en Software Engineering, editores M. Dorfman y R.H. Thayer, IEEE Computer Society Press, 1996, pp.44-54. Reimpreso en "Software Requirements Engineering", editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.73-83.

[Faulk 92] Faulk, S., et al., "The Core Method for Real-Time Requirements", IEEE Software, Vol.9, Nº5, Septiembre 1992, pp.22-33.

[Faulk 96] Faulk, S.R., "Software Requirements: A Tutorial", en Software Engineering, editores M. Dorfman y R.H. Thayer, IEEE Computer Society Press, 1996, pp.82-103. Reimpreso en "Software Requirements Engineering", editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.128-149.

[Finkelstein 96] Finkelstein, A., Dowell, J., "A comedy of Errors: The London Ambulance Service Case Study", IWSSD'96, 8th international Workshop on Software Specification and Design, IEEE Computes Society Press, Alemania, 1996.

[Finkelstein 01] Finkelstein, A., Savigni, A., "A Framework for Requirements Engineering for Context-Aware Services", en anales de 1st. International Workshop From Software Requirements to Architectures (STRAW 01), Toronto, Canadá, Mayo 2001.

[Fiorini 00] Fiorini, S.T., Leite, J.C.S.P., Lucena, C.J.P., "Reusing Process Patterns", 2nd International Workshop on Learning Software Organizations (LSO2000), 2., Oulu-Finlandia, Junio 2000, pp.19-37.

[Firesmith 94] Firesmith, D. G, "Modeling the Dynamic Behavior of Systems, Mechanisms, and Classes with Scenarios", Report on Object Analysis and Design, Vol.1, Nº2, 1994, pp.32-36.

[Floyd 84] Floyd, C., "A systematic look at prototyping", en el libro Approaches to Prototyping, editor R. Buddle, Springer-Verlag, Berlín, Alemania, 1984.

[Fowler 83] Fowler, H.W., "Oxford Fowler's Modern English Usage Dictionary", editor Ernest Gowers, Oxford University Press, Gran Bretaña, 2º edición revisada, Abril 1983.

[Fowler 99] Fowler, M., Scott, K., "UML gota a gota", Addison-Wesley Longman, 1999.

[Fowler 99b] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D., "Refactoring: Improving the Design of Existing Code", Addison-Wesley Professional, 1999.

[Fowler www] Fowler, M., "Variations on a Theme of XP", on-line en su página oficial, http://www.martinfowler.com/articles/xpVariation.html accedida el 27-09-2004.

[Franco 92] Franco, A.P.M., Leite, J.C.S.P., "Uma estratégia de Suporte a Engenharia de Requisitos", XIX Seminario Integrado de Software y Hardware, Río de Janeiro, 1992, pp.200-213.

[Fusaro 97] Fusaro, P., Lanubile, F., Visaggio, G., "A Replicated Experiment to Assess Requirements Inspection Techniques", Empirical Software Engineering, Vol.2, Nº1, 1997, pp.30-57.

[Gane 79] Gane, C., Sarson, T., "Structured Systems Analysis: Tools and Techniques", Prentice-Hall, Englewood Cliffs, NJ, 1979.

[GAO 92] US General Accounting Office, "Mission Critical Systems: Defense Attempting to Address Major Software Challenges", GAO/IMTEC-93-13,

Diciembre 1992.

[GAO 94] US General Accounting Office, "New Denver Airport: Impact of the Delayed Baggage System", GAO/RCED-95-35BR, Octubre 1994, http://ntl.bts.gov/DOCS/rc9535br.html

[García 99] García, O., Gentile, C.G., "Diseño de una herramienta para construcción de LEL y Escenarios", Disertación de Graduación, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina, 1999.

[Garner 98] Garner, B.A., "A Dictionary of Modern American Usage", Oxford University Press, Gran Bretaña, Diciembre 1998.

[Gause 89] Gause, D.C., Weinberg, G.M., "Exploring Requirements. Quality Before Design", Dorset House Publishing, Nueva York, 1989.

[Gibbs 94] Gibbs, W.W., "Trends in Computing: Software's Chronic Crisis", Scientific American, Septiembre 1994, pp.72-81.

[Gilb 89] Gilb, T., "Principles of Software Engineering Management", Addison-Wesley Longman, 1989.

[Gilb 93] Gilb, T., Graham, D., "Software Inspection", Addison-Wesley Publishing Company, 1993.

[Glinz 95] Glinz, M., "An Integrated Formal Model of Scenarios Based on Statecharts", Proceedings of the 5th ESEC, Lecture Notes on Computer Science 989, Springer, Berlin, 1995, pp. 254-271.

[GMoD 92] Germany Ministry of Defense, "V-Model: Software Lifecycle Process Model", General Reprint Nº 250, 1992.

[Godel 62] Godel, K. "On Formally Undecidable Propositions of Principia Mathematica and Related Systems", Dover, Nueva York, 1962.

[Goedicke 99] Goedicke, M., Meyer, T., Taentzer, G., "ViewPoint-oriented Software Development by Distributed Graph Transformation: Towards a Basis for Living with Inconsistencies", IEEE International Symposium on Requirements Engineering, RE'99, Limerick-Irlanda, 1999, pp.92-99.

[Goguen 93] Goguen, J.A., Linde, Ch., "Techniques for Requirements Elicitation", IEEE First International Symposium on Requirements Engineering, RE'93, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp.152-164.

[Gomaa 81] Gomaa, H., Scott, D.B., "Prototyping as a Tool in the Specification of User Requirements", Fifth International Conference on Software Engineering, IEEE Computer Society Press, Marzo 1981, pp.333-342.

[Gotel 94] Gotel, O.C.Z., Finkelstein, A.C.W., "An analysis of the requirements traceability problem", ICRE'94, First IEEE International Conference on Requirements Engineering, IEEE Computer Society Press, Colorado Springs, Abril 1994, pp.94-101.

[Gottesdiener 99] Gottesdiener, E., "Business Rules as Requirements", Software Development, Vol.7, Nº12, Diciembre 1999.

[Gough 95] Gough, P. A., Fodemski, F. T., Higgins, S. A., Ray, S. J., "Scenarios - An Industrial Case Study and Hypermedia Enhancements", RE95: Proceedings of the International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, California, 1995, pp.10-17.

[Greenspan 82] Greenspan, S., Mylopoulos, J., Borgida, A., "Capturing more world knowledge in the requirements specification", 6th. International Conference on Software Engineering, Tokyo, 1982.

[Greer 05] Greer, D., “Requirements Prioritisation for Incremental and Iterative Development”, en el libro “Requirements Engineering for Sociotechnical Systems”, capítulo VII, Information Science Publishing, Maté & Silva editores, Londres, 2005, pp.100-118.

[Gunther 78] Gunther, R., “Management Methodology for Software Product Engineering”, Wiley-Interscience Publication, John Wiley & Sons, Nueva York, 1978.

[Hadad 97] Hadad, G., Kaplan, G., Oliveros, A., Leite, J.C.S.P., “Construcción de Escenarios a partir del Léxico Extendido del Lenguaje”, XXVI JAIIO - SoST’97 Simposio en Tecnología de Software, Buenos Aires, pp.65-77, 1997.

[Hadad 99] Hadad, G., Kaplan, G., Oliveros, A., Leite, J.C.S.P., "Integración de Escenarios con el Léxico Extendido del Lenguaje en la elicitación de requerimientos: Aplicación a un caso real", Revista de Informática Teórica y Aplicada (RITA), Brasil, Vol.6, Nº1, 1999, pp.77-103.

[Hadad 99b] Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., Leite, J.C.S.P., “Enfoque Middle-Out en la Construcción e Integración de Escenarios”, WER’99 - Workshop en Requerimientos, XXVIII JAIIO, Buenos Aires, 1999, pp.79-94.

[Hadad 03] Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., “Construcción de Escenarios Futuros”, anales electrónicos del WICC 2003 - Workshop de Investigadores en Ciencias de la Computación, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Argentina, Mayo 2003.

[Hadad 07] Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., “Creating Software System Context Glossaries”, aceptado para su publicación en Encyclopedia of Information Science and Technology, Idea Group Publishing, 2º edición, 2007.

[Hamilton 91] Hamilton, V.L., Beeby, M.L., “Issues of Traceability in Integrating Tools”, en Tools and Techniques for Maintaining Traceability During Design, IEEE Colloquium, Computing and Control Division, Professional Group C1, Digest Nº180, 1991, pp. 4/1-4/3.

[Hammer 81] Hammer, M., McLeod, D., “Database description with SDM: A semantic data model”, ACM Transactions on Database Systems, Vol.6, Nº3, Septiembre 1981, pp.351-386.

[Hammer 90] Hammer, M., "Reengineering Work: Don't Automate, Obliterate" Harvard Business Review, Julio-Agosto 1990, pp.104-112.

[Harel 87] Harel, D., “Statecharts: a visual formalism for complex systems”, Science of Computer Programming, Vol.8, 1987, pp.231-274.

[Harwell 93] Harwell, R., Aslaksen, E., Hooks, I., Mengot, R., Ptack, K., “What is a Requirement?”, Third International Symposium National Council Systems Engineering, 1993, pp.17-24. Reimpreso en “Software Requirements Engineering”, editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.23-29.

[Haumer 99] Haumer, P., Heymans, P., Jarke, M., Pohl, K., “Bridging the Gap Between Past and Future in RE: A Scenario-Based Approach”, anales de RE’99, International Symposium On Requirements Engineering; 1999 June 7-11; Limerick-Ireland. Los Alamitos, CA: IEEE Computer Society Press, 1999, pp. 66-73.

[Heitmeyer 95] Heitmeyer, C., Gasarch, C., Labaw, B.G., “SCR*: A Toolset for

Specifying and Analyzing Requirements", 10th Annual Conference on Computer Assurance (COMPASS 95), Gaithersburg, MD, USA, Junio 1995, pp.109-122.

[Heitmeyer 96] Heitmeyer, C., Jeffords, R.D., Labaw, B.G., "Automated Consistency Checking of Requirements Specifications", ACM Transactions on Software Engineering and Methodology, Vol.5, Nº3, Julio 1996, pp.231-261.

[Heninger 78] Heninger, K., Parnas, D.L., Shore, J.E., Kallander, J.W., "Software requirements for the A-7E aircraft", Reporte Técnico 3876, Naval Research Laboratory, Washington, 1978.

[Heninger 80] Heninger, K., "Specifying software requirements for complex systems: New techniques and their applications", IEEE TSE, Vol.SE-6, Nº1, Enero 1980, pp.2-13.

[Highsmith 01] Highsmith, J., Cockburn, A., "Agile Software Development: The Business of Innovation", IEEE Computer, Vol.34, Nº9, Septiembre 2001, pp.120-122.

[Highsmith 02] Highsmith, J., "Agile Software Development Ecosystems", Addison-Wesley, 2002.

[Highsmith 02b] Highsmith, J., "What Is Agile Software Development?", CrossTalk - The Journal of Defense Software Engineering, Vol.15, Nº10, Octubre 2002, pp.4-9.

[Hsia 94] Hsia, P., Samuel, J., Gao, J. Kung. D., Toyosima, Y., Chen, C., "Formal Approach to Scenario Analysis", IEEE Software, Vol.11, Nº2, 1994, pp.33-41.

[Hull 05] Hull, E., Jackson, K., Dick, J., "Requirements Engineering", Springer Science, EEUU, 2º edición, 2005.

[Hutchins 92] Hutchins, W.J., Sommers, H.L., "An Introduction to Machine Translation", Academic Press, Londres, 1992.

[Hutchins 98] Hutchins, W.J., "Translation. Technology and the Translator", Machine Translation Review, ISSN 1358-8346, Nº7, Abril 1998, pp.7-14.

[ICONIX 02] "ICONIX QuickStart Plug-In for RUP", ICONIX Software Engineering Inc., http://www.iconixsw.com/, accedida el 25-02-2005.

[IEEE Std 1362-1998] IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document, IEEE, NY, 1998, http://standards.ieee.org/reading/ieee/std_public/description/se/1362-1998_desc.html accedida el 01-04-2005.

[IEEE Std 610.12-1990] IEEE Standard Glossary of Software Engineering Terminology , IEEE, Nueva York, 1990.

[IEEE Std 830-1998] IEEE Recommended Practice for Software Requirements Specifications (ANSI), IEEE, Nueva York, 1998.

[IEEE Std P1233/D3-1995] IEEE Guide for Developing System Requirements for Specifications, IEEE, Nueva York, 1995.

[ISO 9126:2001] International Standard ISO/IEC 9126. Information technology -- Software product evaluation -- Quality characteristics and guidelines for their use, International Organization for Standardization, International Electrotechnical Commission, 2001.

[Jackson 94] Jackson, M., "The Role of Architecture in Requirements Engineering", ICRE'94, First International Conference on Requirements Engineering, IEEE Computer Society, Colorado Springs, Abril 1994, p.241.

[Jackson 95] Jackson, M., "Software Requirements & Specifications. A lexicon of practice, principles and prejudices", Addison-Wesley, Reading, MA/ACM Press, Nueva York, 1995.

[Jacobson 92] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., "Object-Oriented Software Engineering - A Use Case Driven Approach", Reading, MA: Addison Wesley, Nueva York: ACM Press, 1992.

[Jacobson 94] Jacobson, I., "Basic Use-Case Modeling", Report on Object Analysis and Design, Vol.1, Nº2, 1994, pp.15-19.

[Jacobson 94b] Jacobson, I., "Basic Use-Case Modeling (Continued)", Report on Object Analysis and Design, Vol.1, Nº3, 1994, pp.7-9.

[Jacobson 95] Jacobson, I., "The Use Case Construct in Object-Oriented Software Engineering", en "Scenario-Based Design: Envisioning Work and Technology in System Development", J.M. Carroll (ed.), John Wiley and Sons, New York, 1995, pp.309-336.

[Jacobson 99] Jacobson, I., Booch, G., Rumbaugh, J., "The Unified Software Development Process", Addison-Wesley, Reading, MA, 1º edición, Febrero 1999.

[Jarke 98] Jarke, M., "Requirements tracing", Communications of the ACM, Vol.41, Nº12, Diciembre 1998, pp.32-36.

[Jarke 98b] Jarke, M., Bui T.X., Carroll, J.M, "Scenario Management: An Interdisciplinary Approach", Requirements Engineering Journal, Vol.3, N°4, 1998, pp.155-173.

[Johnson 95] Johnson, P., Johnson, H., Wilson, S., "Rapid Prototyping of User Interfaces driven by Task Models", en "Scenario-Based Design: Envisioning Work and Technology in System Development", J.M. Carroll (ed.), John Wiley and Sons, New York, 1995, pp.209-246.

[Jones 84] Jones, T.C., "Reusability in programming: A survey of the state of the art", IEEE TSE, Vol.SE-10, Septiembre 1984, pp.488-494.

[Kaindl 00] Kaindl, H., "A Design Process Based on a Model Combining Scenarios with Goals and Functions", IEEE Transactions on Systems, Man and Cybernetic, Vol.30, Nº 5, Septiembre 2000, pp.537-551.

[Kaindl 02] Kaindl, H., Brinkkemper, S., Bubenko Jr., J.A., Farbey, B., Greenspan, S.J., Heitmeyer, C.L., Leite, J.C.S.P., Mead, N.R., Mylopoulos, J., Siddiqi, J., "Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda", Requirements Engineering Journal, Springer-Verlag London Ltd., Vol.7, Nº3, 2002, pp.113-123.

[Kang 90] Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A., "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report, CMU/SEI-90-TR-21, ADA 235785, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990, http://www.sei.cmu.edu/domain-engineering/domain_eng.html, accedida el 03-10-2005.

[Kantorowitz 97] Kantorowitz, E. Guttman, A., Arzi, L. "The Performance of the N-Fold Inspection Method", Requirements Engineering Journal, Vol.2, Springer-Verlag London, 1997, p.p.152-164,.

[Kaplan 00] Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., Leite, J.C.S.P., "Inspección del Léxico Extendido del Lenguaje", WER'00 – III Workshop de Engenharia de Requisitos, Río de Janeiro, Brasil, Julio 2000, pp.70-91.

[Kaplan 07] Kaplan, G.N., Doorn, J.H., Hadad, G.D.S., "Handling Extemporaneous Information in Requirements Engineering", aceptado para su publicación en Encyclopedia of Information Science and Technology, Idea Group Publishing, 2º edición, 2007.

[Karlsson 98] Karlsson, J., Wohlin, C., Regnell, B., "An evaluation of methods for prioritizing software requirements", Information and Software Technology, Vol.39, Nº14-15, 1998, pp.939-947.

[Kerkow 05] Kerkow, D., Dörr, J., Paech, B., Olsson, T., Koenig, T., "Elicitation and Documentation of Non-Functional Requirements for Sociotechnical Systems", capítulo XVII, Information Science Publishing, Maté & Silva editores, Londres, 2005, pp.284-302.

[Kohler 02] Kohler, K., Paech, B., "Requirement Documents that Win the Race: Not Overweight or Emaciated but Powerful and in Shape", International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Essen, Alemania, 2002, , http://www.agilealliance.org/articles/articles/Kohler.pdf accedido el 28-06-2005.

[Kotonya 93] Kotonya, G., Sommerville, I., "A Framework for Integrating Functional and Non-Functional Requirements", IEEE International Workshop on Systems Engineering for Real Time Applications, Cirencester, UK, 1993, pp.148-153.

[Kotonya 96] Kotonya, G., Sommerville, I., "Requirements Engineering with viewpoints", Software Engineering Journal, IEEE, Vol.11, Nº1, Enero 1996, pp.5-18. Reimpreso en "Software Requirements Engineering", editores Richard H. Thayer y Merlin Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp. 150-163.

[Kotonya 98] Kotonya, G., Sommerville, I., "Requirements Engineering: Process and Techniques", John Wiley & Sons, 1998.

[Kovitz 98] Kovitz, B.L., "Practical Software Requirements: A manual of Content and Style", Greenwich, CT: Manning Publications Co., Octubre 1998.

[Kovitz 02] Kovitz, B.L., "Hidden Skills that Support Phased and Agile Requirements Engineering", International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Essen, Alemania, 2002, http://www-di.inf.puc-rio.br/~julio/tcre-site/p10.pdf y http://www.agilealliance.org/articles/articles/Kovitz.pdf accedidos el 28-03-2005.

[Kruchten 04] Kruchten, P., "The Rational Unified Process: An Introduction", Addison-Wesley, 3º edición, 2004.

[Kyng 95] Kyng, M., "Creating Contexts for Design", en el libro Scenario-Based Design: Envisioning Work and Technology in System Development, editor J. Carroll, John Wiley & Sons, Nueva York, 1995, pp.85-107.

[Laitenberger 00] Laitenberger, O., Debaud, J.M., "An Encompassing Life-Cycle Centric Survey of Software Inspection", Journal of Systems and Software, Vol.50, Nº1, 2000, pp.5-31.

[Lano 80] Lano, R. J., "A Structured Approach for Operational Concept Formulation", TRW SS-80-02, TRW Defense and Space Systems Group, Redondo Beach, California, Enero 1980.

[Larman 03] Larman,C., Basili, V.R., "Iterative and Incremental Development: A Brief History", IEEE Computer, ISSN: 0018-9162, Junio 2003, pp.47-56.

[Leffingwell 00] Leffingwell, D., Widrig, D., "Managing Software Requirements -

A unified approach", Addison-Wesley Longman, 2000.

[Lehman 80] Lehman, M.M., "Programs, Life Cycles, and Laws of Software Evolution", IEEE Special Issue on Software Engineering, Vol.68, Nº9, Septiembre 1980, pp.1060-1076.

[Lehman 85] Lehman, M.M., Belady, L.A., "Program Evolution: processes of software change", Academic Press, Londres, 1985.

[Lehman 91] Lehman, M.M., "Software Engineering: the Software Process and their support", IEEE Software Engineering Journal, IEEE Computer Society Press, Londres, Vol.6, 1991, pp.243-258.

[Leite 89] Leite, J.C.S.P., "Application Languages: A Product of Requirements Analysis", Computer Science Department of PUC-Rio, Brasil, 1989.

[Leite 90] Leite, J.C.S.P., Franco, A.P.M., "O Uso de Hipertexto na Elicitaçao de Linguagens da Aplicaçao", IV Simpósio Brasilero de Engenharia de Software, SBC, Brasil, 1990, pp.134-149.

[Leite 91] Leite, J.C.S.P., Souza, A.L.M.F., "Re-engenheria de Software, um Novo Enfoque para um Velho Problema", XXIV Congresso Nacional de Informática, SUCESU-SP, Brasil, 1991.

[Leite 91b] Leite, J.C.S.P., Freeman, P.A., "Requirements Validation Through Viewpoint Resolution", IEEE Transactions on Software Engineering, Vol. 17, N° 12, pp.1253-1269, 1991.

[Leite 93] Leite, J.C.S.P., Franco, A.P.M., "A Strategy for Conceptual Model Acquisition", IEEE First International Symposium on Requirements Engineering, RE'93, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp.243-246.

[Leite 93b] Leite, J.C.S.P., "Eliciting Requirements Using a Natural Language Based Approach: The Case of the Meeting Scheduler", Monografía en Ciência da Computaçao Nº 13/93, Computer Science Department of PUC-Rio, Brasil, 1993.

[Leite 94] Leite, J.C.S.P., "Engenharia de Requisitos", Notas Tutoriales, material de enseñanza en el curso Requirements Engineering, Computer Science Department of PUC-Rio, Brasil, 1994.

[Leite 95] Leite, J.C.S.P., Oliveira, A.P.A., "A Client Oriented Requirements Baseline", Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, 1995, pp.108-115.

[Leite 97] Leite, J.C.S.P., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., Oliveros, A., "Enhancing a Requirements Baseline with Scenarios", Requirements Engineering Journal, Springer-Verlag London Ltd., Vol.2, N°4, 1997, pp.184-198.

[Leite 97b] Leite, J.C.S.P., "Software Evolution, The Requirements Engineering View", keynote address en anales de XXVI JAIIO - SoST'97 Simposio en Tecnología de Software, Buenos Aires, Argentina, Agosto 1997, pp.21-23.

[Leite 98] Leite J.C.S.P, Leonardi, M.C., "Business rules as organizational Policies", IEEE Ninth International Workshop on Software Specification and Design, IEEE Computer Society Press, 1998, pp.68-76.

[Leite 99] Leite, J.C.S.P., "Anchoring the Requirements Process on Vocabulary", Requirements Capture, Documentation and Validation, E. Börger, B. Hörger, D. Parnas, D. Rombach (editores), Dagstuhl-Seminar 99241, Reporte Nº 242, Junio 1999, pp.13-14.

[Leite 00] Leite, J.C.S.P., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N., "A

Scenario Construction Process", Requirements Engineering Journal, Springer-Verlag London Ltd., Vol.5, Nº1, 2000, pp. 38-61.

[Leite 01] Leite, J.C.S.P., "Extreme Requirements (XR)", Jornadas de Ingeniería de Requisitos Aplicada, Universidad de Sevilla, España, Junio 2001, http://www.lsi.us.es/~amador/JIRA/Ponencias/ JIRA_Leite.pdf.

[Leite 01b] Leite, J.C.S.P., "Gerenciando a Qualidade de Software com Base em Requisitos", en el libro Qualidade de Software: Teoria e Prática, editores A.R.Rocha, J.C. Maldonado y K.C. Weber, Prentice-Hall, San Pablo, capítulo 17, 2001, pp.238-246.

[Leite 03] Leite, J.C.S.P., Doorn, J.H, Hadad, G.D.S., Kaplan, G.N., "Using Scenario Inspections on Different Scenarios Representations", Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, N.33/03, 2003.

[Leite 04] Leite, J.C.S.P., Doorn, J.H., "Perspectives on Software Requirements: An introduction", en el libro "Perspectives on Software Requirements", Kluwer Academic Publishers, EEUU, capítulo 1, 2004, pp.1-5.

[Leite 04b] Leite, J.C.S.P., Doorn, J.H., Kaplan, G.N., Hadad, G.D.S., Ridao, M.N., "Defining System Context using Scenarios", en el libro "Perspectives on Software Requirements", Kluwer Academic Publishers, EEUU, ISBN: 1-4020-7625-8, capítulo 8, 2004, pp.169-199.

[Leite 04c] Leite, J.C.S.P., "Requirements for the agent-oriented paradigm", Panel 1 del VII Workshop on Requirements Engineering (WER'04), Tandil, Buenos Aires, Diciembre 2004. Impreso en la revista Journal of Computer Science & Technology, Universidad Nacional de La Plata, Vol.5, Nº2, Agosto 2005, pp.111.

[Leite 05] Leite, J.C.S.P., Doorn, J.H., Hadad, G.D.S., Kaplan, G.N., "Scenario Inspections", Requirements Engineering Journal, Vol.10, Nº 1, Springer-Verlag London Ltd., Gran Bretaña, Enero 2005, pp.1-21.

[Leite 06] Leite, J.C.S.P., "Gerência dos Requisitos X Gerência por Requisitos", febrero 2006, http://jcspl.wordpress.com/2006/02/20/gerencia-dos-requisitos-x-gerencia-por-requisitos/, accedida el 20-11-07.

[Leite 07] Leite, J.C.S.P., Moraes, E.A., Castro, C.E.P.S., "A Strategy for Information Sources Identification", anales de X Workshop on Requirements Engineering (WER'07), Toronto, Canadá, Mayo 2007, pp.25-34.

[Leonardi 01] Leonardi, M.C., "Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural", M. Sc. Dissertation, Computer Science Department, Universidad Nacional de La Plata, Argentina, Noviembre 2001,128 páginas.

[Lientz 78] Lientz, B.P., Swanson, E.B., "Characteristics of Application Software Maintenance", Communications of the ACM, Vol.21, Nº6, Junio 1978, pp.466-481.

[Lindstrom 93] Lindstrom, D.R., "Five ways to Destroy a Development Project", IEEE Software, Vol.10, Nº5, 1993, pp.55-58.

[Loucopoulos 95] Loucopoulos, P., Karakostas, V., "System Requirements Engineering", McGraw-Hill, Londres, 1995.

[Lubars 93] Lubars, M., Potts, C., Richter, C., "A Review of the State of the Practice in Requirements Modeling", IEEE First International Symposium on Requirements Engineering, RE'93, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp.2-14.

[Lutz 93] Lutz, R., "Analyzing Sotware Requirements Errors in Safety-Critical Embedded Systems", IEEE First International Symposium on Requirements Engineering, RE'93, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp.126-133.

[Macaulay 93] Macaulay, L., "Requirements capture as a cooperative activity", IEEE International Symposium on Requirement Engineering, IEEE Computer Society Press, San Diego, CA, pp.174-181.

[Madhavji 97] Madhavji, N.H., "Impact of Environmental Evolution on Requirements Changes", panel en RE'97, Third IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Enero 1997, pp.98-99.

[Malinowski Rubio 01] Malinowski Rubio, M. P., "Cuando hay que traducir del español al español", VII Simposio Internacional de Comunicación Social, Cuba, 2001, pp. 491-494.

[Martin 90] Martin, J., "Information Engineering", Vols.1-3, Englewood Cliffs, NJ, Prentice Hall, 1990.

[Martin 90b] Martin, J., Tsai, W.T., "N-fold Inspection: A Requirements Analysis Technique", Communications of the ACM, Vol.33, Nº2, 1990, pp.225-232.

[Martin 91] Martin, J., "Rapid Application Development", Macmillan Publishing, Nueva York, 1991.

[Maté 05] Maté, J.L., Silva, A., "Requirements Engineering for Sociotechnical Systems", prefacio, Information Science Publishing, Maté & Silva editores, Londres, 2005.

[Mauco 00] Mauco, M.V., George, C., "Using Requirements Engineering to Derive a Formal Specification", Reporte Técnico Nº 223, UNU/IIST, P.O. Box 3058, Macau, Diciembre 2000.

[Mauco 01] Mauco, M.V., Riesco, D., George, C., "Heuristics to Structure a Formal Specification in RSL from a Client-oriented Technique", 1st Annual International Conference on Computer and Information Science (ICIS'01), ISBN: 0-9700776-2-9, Florida, EEUU, Octubre 2001.

[McCracken 82] McCracken, D.D., Jackson, M.A., "Life Cycle Concept Considered Harmful", Software Engineering Notes, ACM, Abril 1982, pp.29-32.

[McMenamin 84] McMenamin, S.M., Palmer, J.F., "Essential Systems Analysis", Nueva York, Jourdon Press, 1984.

[MÉTRICA v.3] "MÉTRICA: Metodología de Planificación, Desarrollo y Mantenimiento de Sistemas de Información", Ministerio de Administraciones Públicas, http://www.csi.map.es/, España, versión 3, 2000.

[Miller 98] Miller, J., Wood, M., Roper, M., "Further Experiences with Scenarios and Checklists", Empirical Software Engineering, Vol.3, Nº1, 1998, pp.37-64.

[Mills 80] Mills, H.D., O'Neill, D., et al., "The management of software engineering", IBM Systems Journal, Vol.24, Nº2, 1980, pp.414-477.

[Mills 87] Mills, H.D., Dyer, M., Linger, R.C., "Cleanroom Software Engineering", IEEE Software, Vol.4, Nº5, Septiembre 1987, pp.19-25.

[Mizuno 83] Mizuno, Y., "Software Quality Improvement", IEEE Computer, Vol.16, Nº3, 1983, pp.66-72.

[MSF v.3] "MSF: Microsoft Solutions Framework", Microsoft Corporation,

versión 3.0, http://www.microsoft.com/msf.

[Mylopoulos 90] Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M., "Telos: Representing Knowledge about Information Systems", ACM Transactions on Information Systems, Vol.8, Nº4, 1990, pp.327-362.

[Mylopoulos 92] Mylopoulos, J., Chung, L., Nixon, B.A., "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", IEEE TSE, Vol.18, Nº6, 1992, pp.483-497.

[Mylopoulos 98] Mylopoulos, J., "Information Modeling in the Time of The Revolution", Information Systems, Elsevier Science Ltd., Gran Bretaña, Vol.23, Nº3/4, 1998, pp.127-155.

[Mylopoulos 03] Mylopoulos, J., Castro, J., Kolp, M., "Tropos: A Framework for Requirements-Driven Software Development", en el libro Information Systems Engineering: State of the Art and Research Themes, editores J. Brinkkemper y A. Solvberg, Lecture Notes in Computer Science, Springer-Verlag, Junio 2000, pp.261-273.

[Naur 68] Naur, P., Randell, B, editores. "Software Engineering: Report on a Conference Sponsored by the NATO Science Comission", Garmishch, Alemania, Octubre 1968. Scientific Affairs Division, NATO, Bruselas, Enero 1969.

[NCC 87] National Computing Centre (NCC), "The STARTS Guide: A Guide to Methods and Software Tools for the Construction of Large Real-Time Systems", Manchester, 1987.

[Neill 03] Neill, C.J., Laplante, P.A., "Requirements Engineering: The State of the Practice", IEEE Sotware, Vol.20, Nº6, Noviembre/Diciembre 2003, pp.40-45.

[Nuseibeh 94] Nuseibeh, B., Kramer, J., Finkelstein, A., "A Framework for Expressing the Relationship between Multiple Views in Requirements Specification", IEEE TSE, Vol.20, Nº10, 1994, pp.760-773.

[Nuseibeh 00] Nuseibeh, B., Easterbrook, S., "Requirements engineering: a roadmap", International Conference on Software Engineering, ICSE 2000, Future of SE Track 2000, 2000, pp.35-46.

[Oberg 98] Oberg, R., Probasco, L, Ericsson, M., "Applying Requirements Management with Use Cases", Rational Software Corporation, 1998.

[Oliveira 98] Oliveira, A.P.A, "SERBAC – Un método para a definição de requisitos de sistemas", WER'98 Workshop de Engenharia de Requisitos, Maringá, Paraná, Brasil, 1998, pp.108-118.

[Oliveros 04] Oliveros, A., "Requirements vs Analysis where is the boundary if any?", Panel 2 del VII Workshop on Requirements Engineering (WER'04), Tandil, Buenos Aires, Diciembre 2004. Impreso en la revista Journal of Computer Science & Technology, Universidad Nacional de La Plata, Vol.5, Nº2, Agosto 2005, pp.108.

[Paech 05] Paech, B., Denger, C., Kerkow, D., von Knethen, A., "Requirements Engineering for Technical Products: Integrating Specification, Validation and Change Management", capítulo X, Information Science Publishing, Maté & Silva editores, Londres, 2005, pp.153-169.

[Paetsch 03] Paetsch, F., Eberlein, A., Maurer, F., "Requirements Engineering and Agile Software Development", 12th IEEE International Workshops on Enabling Technologies (WETICE 2003), Infrastructure for Collaborative Enterprises, IEEE Computer Society , Linz, Austria, Junio 2003, pp.308-313.

[Palmer 96] Palmer, J.D., "Traceability", en Software Engineering, editores M. Dorfman y R.H. Thayer, IEEE Computer Society Press, 1996, pp.266-276. Reimpreso en "Software Requirements Engineering", editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.364-374.

[Parianou 01] Parianou, A., Kelandrias, P.I., "Interactions between common and specialized language and translation problems of the new scientific terms (with examples from German, English and Greek)", VII Simposio Internacional de Comunicación Social, Cuba, 2001, pp.126-131.

[Parianou 01b] Parianou, A., Kelandrias, P.I., "Neologisms in LSP of Greek/Latin origin and their translation from English/German into Greek", 13th European Symposium on Language for Special Purposes, Vaasa/Vasa - Finlandia, Agosto 2001.

[Parnas 85] Parnas, D.L., Weiss, D, "Active Design Reviews: Principles and Practices", 8º International Conference on Software Engineering, 1985, pp.132-136.

[Parnas 87] Parnas, D.L., "Active Design Reviews: Principles and Practice", Journal of Systems and Software, Vol.7, 1987, pp.259-265.

[Paulk 93] Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, V., "Capability Maturity Model for Software", versión 1.1, Software Engineering Institute, CMU/SEI-93-TR-24, Febrero 1993.

[Peterson 77] Peterson, J., "Petri-nets", ACM Computer Survey, Vol.9, Nº3, Septiembre 1977, pp.223-252.

[Petri 73] Petri, C.A., "Concepts of Net Theory", Mathematical Foundations of Computer Science: Symposium and Summer School, High Tatras, Math. Institute of the Slovak Academy of Sciences, Septiembre 1973, pp.137-146.

[Pigoski 96] Pigoski, T.M., "Practical Software Maintenance", Wiley Computer Publishing, 1996.

[Pimenta 97] Pimenta, M.S., Faust, R., "Eliciting Interactive Systems Requirements in a Language-Centred User-Designer Collaboration: A Semiotic Approach", en Special Issue on HCI and Requirements of ACM SIGCHI Bulletin, Vol.29, Nº1, Enero 1997, pp.61-65.

[Pinheiro 96] Pinheiro, F.A.C., Goguen, J.A., "An object-oriented tool for tracing requirements", IEEE Software, Special issue of papers from ICRE'96, Vol.13, Nº2, Marzo 1996, pp.52-64.

[Pinheiro 02] Pinheiro, F., "Requirements Honesty", International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Essen, Alemania, 2002, http://www-di.inf.puc-rio.br/~julio/tcre-site/p3.pdf accedido el 28-03-2005.

[Pinheiro 04] Pinheiro, F.A.C., "Requirements Traceability", en el libro "Perspectives on Software Requirements", Kluwer Academic Publishers, Estados Unidos, ISBN: 1-4020-7625-8, capítulo 5, 2004, pp.91-113.

[Pitts 99] Pitts, M.G., "The use of evaluative stopping rules in information requirements determination: An empirical investigation of systems analyst behavior", Ph.D. Thesis, Department of Information Systems, University of Maryland, Baltimore, Abril 1999.

[Pitts 04] Pitts, Mitzi G., Browne, G.J., "Stopping Behavior of Systems Analysts During Information Requirements Elicitation", Journal of Management Information Systems, Vol. 21, No.1, 2004 , pp. 203-226.

[Porter 95] Porter, A.A., Votta, Jr., L.G., Basili, V.R., "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment", IEEE TSE, Vol.21, Nº6, 1995, pp.563-575.

[Porter 98] Porter, A.A., Votta, Jr., L.G., "Comparing Detection Methods for Software Requirements Inspections: A Replication Using Professional Subjects", Empirical Software Engineering, Vol.3, Nº4, 1998, pp.355-380.

[Potts 94] Potts, C., Takahashi, K., Antón, A. I., "Inquiry-Based Requirements Analysis", IEEE Software, Vol.11, N°2, 1994, pp.21-32.

[Potts 95] Potts, C., "Using Schematic Scenarios to Understand User Needs", DIS'95 - Symposium on Designing Interactive Systems: Processes, Practices and Techniques, ACM Press, University of Michigan, 1995, pp.247-256.

[Potts 99] Potts, C., "ScenIC: A Strategy for Inquiry-Driven Requirements Determination", International Symposium On Requirements Engineering (RE´99), Limerick, Irlanda, IEEE Computer Society Press, 1999, pp.58-65.

[Prahalad 02] Prahalad, C.K., Krishnan, M.S., "The Dynamic Synchronization of Strategy and Information Technology", MIT Sloan Management Review, Vol.43, Nº4, 2002, pp.23-33.

[Prowell 99] Prowell, S.J., Trammell, C.J., et al., "Cleanroom Software Engineering: Technology and Process", Reading, MA: Addison-Wesley Longman, 1999.

[Ravid 00] Ravid, A., Berry, D. M., "A Method for Extracting and Stating Software Requirements that a User Interface Prototype Contains", Requirements Engineering Journal, Springer-Verlag London Ltd., Vol.5, Nº4, 2000, pp.225-241.

[Regnell 95] Regnell, B., Kimbler, K., Wesslen, A., "Improving the Use Case Driven Approach to Requirements Engineering", Second IEEE International Symposium on Requirements Engineering, Inglaterra, pp. 40-47, Marzo 1995.

[Regnell 96] Regnell, B., "Hierarchical Use Case Modeling for Requirements Engineering", Department of Communication Systems, Lund Institute of Technology, Lund University, Sweden, 1996.

[Regnell 96b] Regnell, B., Anderson, M., Bergstrand, J., "A Hierarchical Use Case Model with Graphical Representation", in Proceedings of ECBS'96, IEEE International Symposium and Workshop on Engineering of Computer-Based Systems, Alemania, 1996, pp.270-277.

[Regnell 99] Regnell, B., "Requirements Engineering with Use Cases – a Basis for Software Development", Ph.D. Thesis, Reporte Técnico 132, Introducción, Department of Communication Systems, Lund University, 1999, pp.7-42.

[Regnell 99b] Regnell, B., Kimbler, K., Wesslén, A., "Improving the Use Case Driven Approach to Requirements Engineering", RE'95, 2nd IEEE International Symposium on Requirements Engineering, Marzo 1995. Forma parte de Ph.D. Thesis: Requirements Engineering with Use Cases – a Basis for Software Development, Reporte Técnico 132, Paper I, Department of Communication Systems, Lund University, 1999, pp.43-63.

[Regnell 99c] Regnell, B., Runesom, P., Thelin, T., "Are the Perspectives Really Different? – Further Experimentation on Scenario-Based Reading of

Requirements", Requirements Engineering with Use Cases – a Basis for Software Development, Reporte Técnico 132, Paper V, Department of Communication Systems, Lund University, 1999, pp. 141-180.

[Reubenstein 91] Reubenstein, H. B., Waters, R. C., "The requirements apprentice: Automated assistance for requirements acquisition", IEEE TSE, Vol.17, Nº3, 1991, pp.226-240.

[Robertson 95] Robertson, S.P., "Generating Object-Oriented Design Representations via Scenario Queries", Scenario-Based Design: Envisioning Work and Technology in System Development, editor J. Carroll, John Wiley & Sons, Nueva York, 1995, pp.279-306.

[Robertson 99] Robertson, S., Robertson, J., "Mastering the Requirements Process", Addison-Wesley Professional, 1º edición, Agosto 1999, 416 páginas.

[Robertson 01] Robertson, J., Robertson, S., "Volere Requirements Specification Template", The Atlantic Systems Guild Inc., edición 9, Julio 2003, http://www.volere.co.uk/

[Robinson 04] Robinson, W., "Surfacing Requirements Interactions", en el libro "Perspectives on Software Requirements", Kluwer Academic Publishers, Estados Unidos, ISBN: 1-4020-7625-8, capítulo 4, 2004, pp.69-90.

[Rockart 96] Rockart, J.F., Earl, M.J., "Eight imperatives for the new IT organization", MIT Sloan Management Review, Vol.38, Issue 1, 1996, pp.43-55.

[Rolland 98] Rolland, C., Souveyet, C, Ben Achour, C., "Guiding Goal Modeling Using Scenarios", IEEE TSE, Vol.24, N°12, 1998, pp.1055–1071.

[Rolland 98b] Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P., "A Proposal for a Scenario Classification Framework", Requirements Engineering Journal, Springer-Verlag London Ltd., Vol.3, Nº1, 1998, pp.23-47.

[Rolland 98c] Rolland, C., Ben Achour, C., "Guiding the construction of textual use case specifications", Data & Knowledge Engineering 25, 1998, pp.125-160.

[Rolland 99] Rolland, C., Grosz, G., Kla, R., "Experience with goal-scenario coupling in Requirements Engineering", anales de RE'99, International Symposium On Requirements Engineering; 1999 June 7-11; Limerick-Irlanda. Los Alamitos, CA: IEEE Computer Society Press, 1999, pp.74–81.

[Rosenberg 99] Rosenberg, D., Scott, K., "Use Case Driven Object Modeling with UML: A Practical Approach", Addison Wesley Object Technology Series, Addison-Wesley Profesional, 1º edición, 1999.

[Ross 77] Ross, D., "Structured Analysis: A Language for Communicating Ideas", IEEE TSE, Vol.SE-3, Nº1, 1977.

[Ross 77b] Ross, D., Schoman, A., "Structured analysis for requirements definition", IEEE TSE, Special Issue on Requirements Analysis, Vol.3, Nº1, 1977, pp. 6-15.

[Ross 97] Ross, R., "The Business Rule Book: Classifying, Defining and Modeling Rules", Business Rule Solutions, LLC, 2º edición, 1997.

[Rout 95] Rout, T., "SPICE: A framework for software process assessment", Software-Process-Improvement and Practice, Pilot Issue, pp.57-66, 1995.

[Royce 70] Royce, W.W., "Managing the Development of Large Software Systems: concepts and techniques", IEEE WESCON, Los Angeles, CA, Agosto 1970.

[Rubin 92] Rubin, K.S., Goldberg, J., "Object Behavior Analysis", Communications of the ACM, Vol.35, N°9, 1992.

[Rumbaugh 91] Rumbaugh, J., M. Blaha, W. Premlani, F. Eddy, W. Lorensen, "Object-Oriented Modeling and Design", Englewood Cliffs, NJ: Prentice Hall International, 1991.

[Rumbaugh 94] Rumbaugh, J., "Getting Started: Use Cases to Capture Requirements", Journal of Object Oriented Programming, Vol.23, Septiembre 1994, pp.8-12. Reimpreso en "Software Requirements Engineering", editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.123-127.

[Rumbaugh 05] Rumbaugh, J., Jacobson, I., Booch, G., "The Unified Modeling Language Reference Manual", Addison-Wesley, Reading, MA, 2º edición, 2005.

[RUP 98] Rational Software, "RUP: Rational Unified Process", IBM Corporation, 1998, http://www-306.ibm.com/software/rational, accedida el 24-02-2005.

[Rus 02] Rus, I., Lindvall, M., "Knowledge Management in Software Engineering", IEEE Software, Vol.19, Nº3, 2002, pp.26-38.

[Saaty 80] Saaty, T.L., "The Analytic Hierarchy Process", McGraw-Hill, 1980.

[Sábat Neto 00] Sábat Neto, J.M., "Integrando Requisitos Não Funcionais à Modelagem Orientada a Objetos", M.Sc. Dissertation, Computer Science Department of PUC-Rio, Brasil, 2000, 206 páginas.

[Sandall 98] Sandall, K., Blomkvist, O., Karlsson, J., Krysander, C., Lindvall, M., Ohlsson, N., "An Extended Replication of an Experiment for Assessing Methods for Software Requirements", Empirical Software Engineering, Vol.3, Nº4, 1998, pp.381-406.

[Saroka 02] Saroka, R., "Sistemas de Información en la Era Digital", Fundación OSDE, Buenos Aires, 2002.

[Sayão 05] Sayão, M., Leite, J.C.S.P. "Rastreabilidade de Requisitos", Monografias em Ciência da Computação, No. XX/05, ISSN: 0103-9741, PUC-Rio, Brazil, mayo 2005.

[Sawyer 01] Sawyer, P., Kotonya, G., "Software Requirements", SWEBOK, Guide to the Software Engineering Body of Knowledge, editores P. Bourque y R. Dupuis, IEEE Computer Society, Los Alamitos, CA, capítulo 2, 2001, pp.31-56, IEEE Trial Version 1.00, http://www.swebok.org/stoneman/trial_1_00.html, accedida el 20-02-2005.

[Scalzo 95] Scalzo, B., "UPROAR – User Processes Reveal Objects And Requirements", OOPSLA'95, Workshop on Use Cases, 1995.

[Schmuller 00] Schmuller, J., "Aprendiendo UML en 24 horas", Pearson Educación Latinoamérica, 2000.

[Schneider 92] Schneider, G., Martin, J., Tsai, W.T., "An experimental study of fault detection in user requirements documents", ACM Transaction on Software Engineering and Methodology, Vol.1, Nº2, 1992, pp.188-204.

[Schneider 98] Schneider, G., Winters, J., "Applying Use Cases, A Practical Guide", Addison-Wesley, Reading, MA, 1998.

[Scott 00] Scott, J.A., Nisse, D., "Software Configuration Management", SWEBOK, Guide to the Software Engineering Body of Knowledge, IEEE-Stoneman, capítulo 7, versión 0.7, Abril 2000, pp. 7.1-7.18,

http://www.swebok.org, acedida el 25-08-2005.

[Senn 89] Senn, J.A., "Analysis & Design of Information Systems", 2º edición, Mc Graw-Hill Inc., 1989.

[Shlaer 88] Shlaer, S., Mellor, S., "Object-Oriented System Analysis: Modeling the World in Data", Prentice-Hall, Englewood Cliffs, N.J., 1988.

[Shull 98] Shull, F., "Developing Techniques for Using Software Documents: A Series of Empirical Studies", PhD Thesis, Computer Science Department, University of Maryland, EEUU, 1998.

[Siddiqi 96] Siddiqi, J., Shekaran M.C., "Requirements Engineering: The Emerging Wisdom", IEEE Software, Vol.13, Nº2, Marzo 1996, pp.15-19. Reimpreso en "Software Requirements Engineering", editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.36-40.

[Simon 69] Simon, H.A., "The Sciences of the Artificial", Cambridge, MA, MIT Press, 1969.

[Sommerville 97] Sommerville, I., Sawyer, P., "Requirements Engineering: A good practice guide", John Wiley & Sons, 1997.

[Sommerville 02] Sommerville, I., "Ingeniería de Software", Addison-Wesley, 6º edición, 2002.

[Spivey 89] Spivey, J., "The Z Notation: A Reference Manual", Prentice-Hall, 1989.

[Stapleton 97] Stapleton, J., "DSDM: Dynamic Systems Development Method", Addison-Wesley, 1997.

[Stix 94] Stix, G., "Trends in Air Transportation: Aging Airways", Scientific American, Mayo 1994, pp.70-78.

[Sutcliffe 97] Sutcliffe, A., "Workshop: Exploring Scenarios in Requirements Engineering", RE'97, Third IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp.180-182.

[Sutcliffe 97b] Sutcliffe, A., "A Technique Combination Approach to Requirements Engineering", RE'97, Third IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, Colorado, 1997, pp.65-74.

[Sutcliffe 98] Sutcliffe, A.G., Maiden, N.A.M., Minocha, S., Manuel, D., "Supporting Scenario-Based Requirements Engineering", IEEE TSE, Vol.24, Nº12, 1998, pp.1072-1088.

[Swartout 82] Swartout, W., Blazer, R., "On the Inevitable Intertwining of Specification and Design", Communications of the ACM, Vol.27, Nº7, Julio 1982, pp.438-440.

[Tavolato 84] Tavolato, P., Vincena, K., "A Prototyping Methodology and its Tool", en Approaches to Prototyping, editor R. Budde, Springer-Verlag, Berlín, 1984, pp.434-446.

[Thayer 97] Thayer R.H., Dorfman, M., "Software Requirements Engineering", IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997.

[Thayer 97b] Thayer R.H., Dorfman, M., "Introduction to Tutorial", en el libro "Software Requirements Engineering", IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.1-2.

[Trujillo 99] Trujillo, A., "Translation Engines: Techniques for Machine Translation", Londres: Springer-Verlag, 1999.

[UML 03] OMG, "Unified Modeling Language Specification", versión 1.5., Marzo

2003, http://www.omg.org/technology/documents/formal/uml.htm

[Ureña 01] Ureña López, L.A., García Vega, M, Martínez Santiago, F., "Explotando las Relaciones Léxicas y Semánticas de WordNet en la Resolución de la Ambigüedad Léxica", VII Simposio Internacional de Comunicación Social, Cuba, 2001, pp.414-418.

[van Lamsweerde 93] van Lamsweerde, A., Darimont, R., Massonet, Ph., "The Meeting Scheduler System - Preliminary Definition", Reporte interno, Université Catholique de Louvain, 1993.

[van Lamsweerde 98] van Lamsweerde, A., "Inferring Declarative Requirements Specifications from Operational Scenarios", IEEE Transactions on Software Engineering, Vol. 24, Nº12, Diciembre 1998, pp.1089-1114.

[van Schouwen 93] van Schouwen, A.J., Parnas, D.L., Madey, J., "Documentation of Requirements for Computer Systems", IEEE International Symposium on Requirements Engineering (RE93), San Diego, California, EEUU, Enero 1993, pp.198-207.

[Vienneau 93] Vienneau, R., "A Review of Formal Methods", Kaman Science Corporation, 1993, pp.3-15 y 27-33. Reimpreso en "Software Requirements Engineering", editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.324-335.

[von Bertalanffy 68] von Bertalanffy, L., "General System Theory: Foundations, Development, Applications", Nueva York, 1968. Edición revisada: 1976, editor George Braziller.

[Wallace 97] Wallace, D.R., Ippolito, L.M., "Verifying and Validating Software Requirements Specifications", impreso en "Software Requirements Engineering", editores R.H. Thayer y M. Dorfman, IEEE Computer Society Press, 2º edición, Los Alamitos, CA, 1997, pp.389-404.

[Ward 85] Ward, P., Mellor, S. "Structured Development for Real-Time Systems", Prentice-Hall, Englewood Cliffs, NJ, 1985.

[Weidenhaupt 98] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P., "Scenarios in System Development: Current Practice", IEEE Software, 1998, pp.34-45.

[Whitenack 94] Whitenack, B.G. Jr., "RAPPeL: A Requirements Analysis Process Pattern Language for Object Oriented Development", Knowledge Systems Corp., 1994.

[Whitten 03] Whitten,J., Bentley,L., Dittman, K., "Systems Analysis and Design Methods", Mc Graw-Hill / Irwin, 6º edición, 2003.

[Wieringa 95] Wieringa, R.J., "An introduction to requirements traceability", Reporte Técnico IR-389, Faculty of Mathematics and Computer Science, University of Vrije, Amsterdam, Septiembre 1995.

[Wirfs-Brock 90] Wirfs-Brock, R., Wilkerson, B., Wiener, L., "Designing Object-Oriented Software", Prentice-Hall, Englewood Cliffs, N.J., 1990.

[Wirfs-Brock 95] Wirfs-Brock, R., "Designing Objects and Their Interactions: A Brief Look at Responsibility-Driven Design", Scenario-Based Design: Envisioning Work and Technology in System Development, editor J. Carroll, John Wiley, Nueva York, 1995, pp. 337-359.

[Wohlin 98] Wohlin, C., Runeson, P., "Defect content estimations from Review Data", 20th International Conference on Software Engineering, Kyoto, Japón, IEEE Computer Society Press, 1998, pp. 400-409.

[Wood 89] Wood, J., Silver, D. "Joint Application Design", John Wiley & Sons,

Nueva York, 1989.

[Yeh 90] Yeh, R.T., Ng, P.A., “Software Requirements – A Management Perspective”, System and Software Requirements Engineering, editores M. Dorfman y R.H. Thayer, IEEE Computer Society Press, 1990, pp.450-461.

[Yen 97] Yen, J., Tiao, W., “A Systematic Tradeoff Analysis for Conflicting Imprecise Requirements”, RE’97, Third IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Enero 1997, pp.87-97.

[Young 04] Young, R.R., “The Requirements Engineering Handbook”, Artech House, Norwood, MA, 2004.

[Yourdon 89] Jourdon, E., “Modern Structured Analysis”, Englewood Cliffs, NJ, Yourdon Press/Prentice Hall, 1989.

[Yourdon 89b] Yourdon, E., “Structured Walkthroughs”, Prentice Hall, 4º edición, N.Y., 1989.

[Yu 93] Yu, E., “Modeling organizations for information systems requirements engineering”, IEEE First International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego, Enero 1993, pp.34-41.

[Yu 95] Yu, E., “Modeling Strategic Relationships for Process Reengineering”, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.

[Yu 97] Yu, E., “Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering”, IEEE International Symposium of Requirements Engineering (RE’97), Annapolis, Maryland, Enero 1997, pp.226-235.

[Zanlorenci 98] Zanlorenci, E.P., Burnett, R.C., “Modelo para Qualificaçao da Fonte de Informaçao do Cliente e de Requisito Funcional”, WER’98, Workshop en Engenharia do Requisitos, Maringá, Brasil, 1998, pp.39-48.

[Zave 84] Zave, P., “The operational versus the conventional approach to software development”, Communications of the ACM, Vol.27, Nº2, 1984.

[Zave 97] Zave, P., Jackson, M., “Four Dark Corners of Requirements Engineering”, ACM Transactions on Software Engineering and Methodology, Vol.6, Nº1, 1997, pp.1-30.

[Zave 01] Zave, P., "Requirements for Evolving Systems: A Telecommunications Perspective”, Fifth IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp.2-9.

[Zorman 95] Zorman, L., “Requirements Envisaging by Utilizing Scenarios (Rebus)”, Ph.D. Dissertation, University of Southern California, 1995.

[Zultner 92] Zultner, R., "Quality Function Deployment (QFD) for Software: Structured Requirements Exploration", en el libro Total Quality Management for Software, editores G. Schulmeyer y J. McManus, Nueva York: Van Nostrand Reinhold, 1992, pp.297-317.

Agradecimientos

Eternamente agradecida a mi familia por su apoyo incondicional, y su entusiasmo aún en los peores momentos. A Ignacio que ilumina mi vida. A Mensario y Elías que son mi vida.

A Jorge H. Doorn, quien ha sido el máximo colaborador en esta tesis, por su paciencia, inteligencia y sus innumerables anécdotas. Parafraseando lo que un amigo de Jorge L. Borges decía de él "que era las 24 horas del día un escritor", yo diría sin equivocarme "que Jorge es las 24 horas del día un docente", sus charlas son siempre clases de temas tan diversos que uno no puede dejar de respirar sin aprender.

A Julio S.P. Leite por su amistad, humanidad y por compartir su conocimiento e ideas con una generosidad incalculable.

Un agradecimiento especial a mi amiga y colega, Gladys N. Kaplan, por su colaboración y participación en tantas charlas y reuniones, y su fervor en el trabajo y por darme ánimo a continuar.

A muchos colegas por su colaboración, entre ellos destaco a Marcela Ridao y al grupo INTIA de la Universidad Nacional del Centro de la Provincia de Buenos Aires.

A mis amigas, algunas de ellas también colegas, les agradezco su apoyo y su presencia cuando más lo necesitaba.

Graciela D. S. Hadad
Noviembre, 2007