

**Universitat Politècnica de Catalunya**  
**Departament de Llenguatges i Sistemes Informàtics**  
**Màster en Computació**

# **Tesis de Máster**

## **Construcción de un catálogo de patrones de requisitos funcionales para ERP**

**Estudiant: María Cristina Carreón Suarez del Real**  
**Directors: Xavier Franch, Carme Quer**  
**Data: 25/06/2008**

## Contenido

1	Introducción.....	5
1.1	Motivación.....	5
1.2	Objetivo .....	6
1.3	Justificación .....	6
1.4	Estructura de la tesis .....	7
2	Estado del arte .....	8
2.1	Selección de componentes OTS .....	8
2.1.1	Clasificación y Definición de componentes OTS.....	9
2.1.2	Ingeniería de software basado en componentes .....	12
2.1.3	Características de la selección de componentes OTS.....	13
2.1.4	Métodos de Selección de componentes OTS .....	16
2.1.5	Selección de componentes en el grupo GESSI.....	18
2.2	Ingeniería de Requisitos.....	19
2.2.1	Definición de Ingeniería de Requisitos .....	20
2.2.2	El ciclo de vida de la ingeniería de requisitos .....	21
2.2.3	Tipos de Requisitos .....	21
2.2.4	Elicitación de Requisitos .....	23
2.2.5	Reutilización.....	24
2.2.6	Requisitos en el grupo GESSI .....	26
2.3	Patrones en Ingeniería de Software .....	27
2.3.1	Historia .....	27
2.3.2	Definición y características .....	28
2.3.3	Patrones en ingeniería de software .....	29
2.3.4	Patrones en ingeniería de requisitos .....	30
2.3.5	Propuestas existentes de uso de patrones de requisitos .....	31
2.4	Calidad de Software.....	33
2.4.1	Enfoques de Calidad de Software.....	33
2.4.2	ISO/IEC 9126-1.....	34
2.4.3	ISO/IEC 25000:2005 .....	38
2.4.4	Calidad de software en el grupo GESSI .....	41
2.5	Ingeniería de Dominio .....	42
2.5.1	Análisis del Dominio .....	44
2.5.2	Modelado Orientado a Objetivos y el Modelado Orientado a Actores .....	45
2.5.3	Aportación de GESSI a $i^*$ .....	51
2.6	ERP .....	52
2.6.1	Historia .....	53
2.6.2	Definición y Características .....	55
2.6.3	Módulos .....	56
2.6.4	Relación entre los ERP y otros sistemas de gestión. ....	60
2.6.5	Selección de ERP .....	61
2.6.6	Contribución del grupo GESSI a la selección de ERP .....	66
3	Construcción de un catálogo de patrones de requisitos funcionales para ERP .....	67
3.1	Definición .....	68

3.1.1	Clasificación de los patrones .....	70
3.2	Consideraciones previas .....	71
3.2.1	CITI y su papel en la selección de ERP .....	72
3.2.2	Colaboración entre CITI y GESSI .....	74
3.3	Método .....	75
3.3.1	Análisis del dominio de ERP .....	76
3.3.2	Método aplicado .....	82
3.4	Diagrama UML para la creación de un patrón .....	92
3.5	Ejemplo de aplicación del método propuesto. ....	93
3.6	Aplicaciones .....	95
4	Conclusiones y líneas futuras .....	95
5	Bibliografía .....	97
7	Apéndice 1 .....	112
8	Apéndice 2 .....	121

## Lista de Tablas

Tabla 2-A:	Actividades los componentes OTS .....	14
Tabla 2-B:	Algunas propuestas existentes sobre el uso de patrones .....	32
Tabla 2-C:	El modelo de calidad interno/externo ISO/IEC 9126-1 .....	37
Tabla 2-E:	Comparativa entre módulos ERP .....	58
Tabla 2-F:	Métodos para la selección de ERP según perspectiva .....	66

## Lista de Figuras

Figura 2-A:	Entradas y salidas del proceso de ingeniería de requisitos .....	20
Figura 2-B:	Ciclo de vida de la ingeniería de requisitos .....	21
Figura 2-C:	Espiral de elicitación análisis y negociación .....	24
Figura 2-D:	Utilización del estándar ISO 9126 .....	34
Figura 2-E:	Partes del estándar ISO/IEC 9126 .....	35
Figura 2-F:	Vistas del estándar ISO/IEC 9126 .....	35
Figura 2-G:	Arquitectura de SQuaRE .....	39
Figura 2-H:	Ciclo de vida de la calidad de un producto de software .....	40
Figura 2-I:	Estructura del modelo de calidad .....	40
Figura 2-J:	Relación de SQuaRE con ISO/IEC 9126 e ISO/IEC 14598 .....	41
Figura 2-K:	Reutilización de Software .....	43
Figura 2-L:	Actividades de la Ingeniería de Dominio .....	44
Figura 2-M:	Tareas relativas al Análisis de Dominio .....	45
Figura 2-N:	Modelo $i^*$ de Yu para un proceso de tutoría académica .....	51
Figura 2-O:	Evolución de los ERP .....	54
Figura 2-P:	Anatomía de un ERP .....	57
Figura 2-Q:	Relación de ERP con otros sistemas .....	61

Figura 2-R: Ciclo de vida de los ERP .....	62
Figura 3-A: Componentes de un patrón .....	69
Figura 3-B: Plantilla de patrones funcionales para ERP .....	70
Figura 3-C: Clasificación de functionality en el ISO/IEC 9126-1. ....	71
Figura 3-D: Modelo de calidad para clasificar patrones de requisitos funcionales para ERP .....	71
Figura 3-E: Forma de trabajo CITI/CASSIS .....	73
Figura 3-F: Método propuesto.....	75
Figura 3-G: Obtención de dependencias entre módulos ERP .....	77
Figura 3-I: Modelo SD $i^*$ . Relación entre módulos básicos de un ERP (excepto Finanzas).....	79
Figura 3-J: Modelo SD $i^*$ . Relación entre módulos básicos de un ERP (Finanzas) .....	80
Figura 3-K: Pasos para la construcción de un catálogo de patrones .....	82
Figura 3-L: Extracción de requisitos funcionales.....	83
Figura 3-O: Refinamiento de catálogo de requisitos funcionales .....	85
Figura 3-Q: Catálogo de candidatos a patrones funcionales (nuevo candidato) .....	86
Figura 3-S: Repositorio de patrones de requisitos funcionales ERP.....	88
Figura 3-T: Partes de un patrón de requisitos.....	89
Figura 3-U: Dependencia entre extensiones.....	89
Figura 3-V: Caso 1. Candidato totalmente contenido en el patrón. ....	90
Figura 3-W: Caso 2. Una parte específica no incluida en el patrón. ....	90
Figura 3-X: Caso 3. Una parte específica sin incluir $F_1$ en el patrón ( $F_1 > E_1$ ). ....	90
Figura 3-Y: Caso 3. Una parte específica sin incluir $F_1$ en el patrón ( $E_1 > F_1$ ). ....	90
Figura 3-Z: Ejemplo de patrón de requisitos funcionales ERP .....	91
Figura 3-AA: Diagrama UML para la creación de un patrón ERP. ....	92
Figura 3-BB: Análisis semántico y refinamiento(requisito 1) .....	93
Figura 3-CC: Catálogo de candidatos a patrón (Candidato nuevo).....	93
Figura 3-DD: Análisis semántico y refinamiento (requisito 2) .....	94
Figura 3-EE: Catálogo de candidatos a patrón (Candidato Asociado).....	94

# 1 Introducción

La presente tesis de máster constituye una primera aproximación a un procedimiento para la construcción de un catálogo de patrones de requisitos funcionales en el dominio de los *sistemas Enterprise Resource Planning (ERP)*. La construcción de dicho catálogo tiene la finalidad de contribuir a mejorar el proceso de selección de ERP mediante patrones que facilitan la reutilización de requisitos. Para llegar a esta propuesta de procedimiento, se han analizado los temas relacionados llegando a un estado del arte de cada uno de ellos, se ha proporcionado una definición de patrón de requisitos funcional, y se ha realizado la extracción de patrones de un conjunto de libros de requisitos obtenidos de proyectos reales de selección de ERP.

## 1.1 Motivación

Debido a las crecientes presiones para obtener software de calidad y en el menor tiempo, desde hace ya bastantes años que se han venido introduciendo técnicas de reutilización en el proceso de desarrollo del software [1]. Estas técnicas facilitan que los componentes se diseñen y desarrollen con el objetivo de poder ser reutilizados en otras aplicaciones, reduciendo el tiempo de desarrollo, mejorando la calidad del producto y siendo más competitivos en costes.

Idealmente la reutilización consiste en hacer uso del conocimiento en su forma más abstracta [2]. Los requisitos representan el nivel más abstracto del conocimiento de un dominio particular por lo que al reutilizar a partir de los requisitos se incrementa el nivel de abstracción de los elementos reutilizables.

Debido a esto, el concepto de reutilización en la etapa de ingeniería de requisitos es bien aceptado como un objetivo deseable desde hace tiempo [3-9]. Se han realizado varias propuestas de técnicas a usar al respecto sin embargo, no parece haber propuestas concretas y lo que parece faltar en la literatura es la forma que deben tener los requisitos para que puedan lograr la reutilización como parte de la práctica regular de proyectos.

Con la motivación de la reutilización de requisitos, de entre todas las técnicas propuestas para alcanzarla, en esta tesis nos centramos en el uso de patrones de requisitos. En el ámbito de la ingeniería de software, los patrones fueron creados, en primera instancia, para resolver problemas identificados durante el diseño de sistemas software, y desde hace ya bastantes años, muchos esfuerzos de investigación y desarrollo en ingeniería de software se han centrado en la identificación y el uso de este tipo de patrones [10]. Sin embargo, en los últimos años, también han aparecido propuestas para que sean usados en otros ámbitos de la ingeniería del software [11-13].

El motivo del uso de patrones, es la percepción que tenemos y que nos gustaría poder validar en un futuro, de que el uso de patrones de requisitos puede ayudar a resolver uno de los problemas que suelen existir en los libros de requisitos. Concretamente, se trata de que en dichos libros habitualmente aparecen definiciones de requisitos ambiguas, incompletas, incoherentes y por lo general están expresadas de forma poco sistemática [14]. El uso de

patrones creemos que permitirá crear una forma estándar para la elaboración y redacción de requisitos. Además de que ayudarán a:

- Expresar los requisitos de forma general y clara.
- Reducir el tiempo y el esfuerzo de elicitación.
- Eliminar requisitos redundantes.
- Mejorar la calidad de los requisitos.
- Mejorar la calidad en la generación de libros de requisitos.
- Promover la reutilización.
- Identificar nuevos requisitos.
- Adaptar los requisitos a nuevas funcionalidades.
- Guiar el proceso de aprendizaje para el conocimiento de un dominio.
- Brindar ahorros económicos.

Para alinear este trabajo con los intereses del grupo GESSI, al que pertenezco, nos centraremos además en la construcción de patrones de requisitos en el ámbito de la selección de componentes software. El grupo GESSI lleva ya casi diez años de investigación en este ámbito, y creemos que, en él, las ayudas proporcionadas por el uso de patrones podrían ser bien explotadas. Ya que realizar una buena elicitación, definición, búsqueda y alineamiento de requisitos es de especial importancia para la adquisición de un producto de calidad y más cuando el producto a adquirir pertenece a dominios de software extensos.

La comunidad de reutilización ha informado el éxito en el uso de enfoques de un dominio específico para su reutilización [15, 16]. Por lo que el desarrollo de esta tesis se centrará en un dominio concreto de software, el dominio de los *Enterprise Resource Planning (ERP)* y más concretamente en los patrones de requisitos funcionales para ERP.

## 1.2 Objetivo

El objetivo de esta tesis de máster es:

*“Realizar un estudio del estado del arte, definir un primer método para la construcción de un catálogo de patrones de requisitos funcionales para la selección de ERP, y crear unas primeras heurísticas de dicho catálogo, partiendo de los requisitos utilizados en diferentes proyectos reales de selección de este tipo de componentes y clasificar dichos patrones usando como criterios de clasificación las características de calidad definidas dentro del estándar ISO 9126-1.”*

## 1.3 Justificación

Por una parte, el dominio de los ERP es de gran importancia en la actualidad, ya que muchas organizaciones confían en este tipo de sistemas para la gestión de la parte básica de su organización. Además, son componentes de gran tamaño y de gran impacto en dichas organizaciones y por tanto requiere de especial atención.

Por otra parte, por no querer abarcar demasiado, y debido a que las organizaciones consideran que la adaptabilidad de los requisitos funcionales es uno de los criterios

más importantes en el proceso de selección y evaluación de un ERP es la funcionalidad [17-19] nos centraremos en los requisitos funcionales. Como es bien sabido, existen otros tipos de requisitos no funcionales, y no técnicos. En este caso y por la naturaleza de los ERP, nos parece especialmente interesante tomar como punto de partida los requisitos funcionales por varios motivos:

Primeramente porque estos requisitos expresan los deseos de la organización acerca de lo que el sistema debe hacer. También porque la funcionalidad es el criterio mas importante para decidir si un elemento reutilizable se puede usar en un contexto dado [20]. Y, finalmente, porque se puede interpretar que los requisitos funcionales son el hilo conductor de la reutilización [21] y el punto de partida para una estrategia completa de articulación de soluciones a partir de elementos reutilizables. Además, el expresar los requisitos de forma clara y estructurada permite que puedan ser reutilizables, ayudando al alineamiento y la identificación del conjunto de requisitos funcionales del ERP con los requisitos funcionales esperados por la organización.

Con la finalidad de clasificar los patrones de requisitos funcionales para ERP, usaremos el modelo de calidad del estándar ISO 9126-1. Creemos que esta clasificación puede facilitar la identificación y estructuración de los patrones durante el proceso de definición de requisitos dentro de un proyecto concreto.

## **1.4 Estructura de la tesis**

En este capítulo se han definido la motivación y la justificación para el desarrollo de la presente tesis de máster, el contexto de trabajo y los objetivos a cumplir. Para los siguientes capítulos la tesis queda estructurada de la siguiente forma: el capítulo 2 presenta el estado del arte de los temas relacionados con esta tesis de máster, el capítulo 3 describe el método realizado para la construcción de un primer catálogo de patrones funcionales para ERP, y finalmente el capítulo 4 agrupa las conclusiones y trabajo previsto para el futuro.

## 2 Estado del arte

En esta sección se presenta el estado del arte de los temas relacionados con el desarrollo de esta tesis de máster. Este estado del arte brinda la teoría necesaria acerca de los conceptos básicos requeridos para entender la definición y la importancia de la creación de un catalogo de patrones de requisitos funcionales de ERP descritos en el apartado 3 y presentados en el anexo 2. Dispone de 6 secciones en las cuales se describe:

- En la sección 2.1 se presenta la definición de componente *off-the-shelf* (OTS) y sus características, importancia y los métodos existentes para la selección.
- En la sección 2.2 se describe la ingeniería de requisitos, los tipos de requisitos y sus principales características, el ciclo de vida de la ingeniería de requisitos, el proceso de elicitación, la reutilización de requisitos y su importancia.
- En la sección 2.3 se presenta la historia, definición y características de los patrones en ingeniería de software, los patrones en ingeniería de requisitos y las propuestas existentes para su utilización.
- En la sección 2.4 se presenta los conceptos referentes a calidad y sus distintos enfoques, se describen el ISO/IEC 9126-1 y el ISO/IEC 25000:2005 y su aplicación
- En la sección 2.5 se describen la ingeniería de dominio, el análisis de dominio y el modelado de dominio orientado a objetivos y a actores, en concreto se analiza el lenguaje *i\**.
- En la sección 2.6 se describe la historia, definición, características y ciclo de vida de los ERP, así como la importancia de su correcta selección y los métodos existentes.
- Además, en cada una de estas secciones se presentan las aportaciones del grupo GESSI al tema respectivo (a excepción de patrones).

### 2.1 Selección de componentes OTS

La reutilización de componentes para construir sistemas más grandes se ha extendido rápidamente, ya que entre otras ventajas, permite reducir costes, disminuir tiempos de desarrollo, y facilitar la adaptación a las nuevas tecnologías.

Sin embargo, también introduce nuevos riesgos y retos. Esto es debido, entre otros factores, a la falta de información (relevante y disponible) para poder realizar una búsqueda eficiente y de calidad que facilite la selección y evaluación de componentes.

En caso de que como resultado de un proyecto de selección, se seleccione un componente inadecuado, el riesgo de que la implantación de dicho componente fracase se incrementa exponencialmente [22].

A continuación exploraremos el estado del arte en la selección de componentes OTS. En la sección 2.1.1 se clasifica y define los componentes OTS. En la sección 2.1.2 se desarrolla el tema de ingeniería de software basado en componentes exponiendo conceptos, ventajas y desventajas. En la sección 2.1.3 se muestra la importancia de la selección de componentes OTS, describiendo además sus fases, funciones y condicionantes. En la sección 2.1.4 se describe los métodos propuestos hasta el momento para la selección de componentes OTS. Por último en la sección 2.1.5 se describe las principales aportaciones realizadas por el grupo GESSI en este ámbito.



### 2.1.1 Clasificación y Definición de componentes OTS

Existen distintas clasificaciones posibles de los componentes software. Una de ellas es la que divide los componentes según si son desarrollados a medida, para satisfacer las necesidades de un cliente específico, o bien son desarrollados para satisfacer las necesidades de múltiples clientes.

Estos últimos, son denominados comúnmente componentes *off-the-shelf* (OTS). Según [23] un *componente OTS* es un subsistema reutilizable, que combina funcionalidades que cumplen con gran parte de los requisitos a nivel de negocio, disponible al público en general con un costo o ciertas obligaciones de licencia. Un componente OTS posee las siguientes características:

- Puede ser utilizado tal "como es"
- Soporta personalización de código, mediante una envoltura de software o mediante una extensión.
- Es ajustable, o parcialmente modificable.
- Es interoperable con otros componentes.
- Está suficientemente documentado.

Los componentes OTS a su vez se pueden clasificar en dos tipos de componentes, descritos en las sub-secciones 2.1.1.1 y 2.1.1.2.

#### 2.1.1.1 Componentes Open Source Software (OSS)

También llamados componentes de software libre. Los componentes OSS son ofrecidos, sin ningún coste, por comunidades de código abierto con código fuente de libre acceso [24].

Para entender el término código abierto y software libre debemos hacer un poco de historia. Open Source (OS) y Free Software (FS) son movimientos distintos, con puntos de vista y objetivos diferentes, que sin embargo tienen puntos comunes y una historia enlazada. Para entender el concepto de OS, hay que entender primero el movimiento FS.

FS fue un movimiento iniciado en 1983 por Richard Stallman, un programador del MIT, con el anuncio del proyecto GNU [25]. Este proyecto tenía como objetivo crear un sistema operativo completo y libre compatible con el sistema operativo UNIX, y crear la Fundación de Software Libre [26] para promoverlo.

En 1998, como reacción al anuncio de la liberación del código fuente del navegador Netscape (conocido como Mozilla), se crea el movimiento OS que nace como una escisión del movimiento FS. A este movimiento se le llamó Open Source Initiative (OSI) [27], y fue liderado por Eric S. Raymond y Bruce Perneer. Este nuevo movimiento adoptó el término código abierto al tratarse de una expresión que es menos ambigua y más cómoda para el mundo empresarial [28]. Hasta el momento el término "free" creaba muchas confusiones por significar tanto "libre" como "gratis". FS y OS describen la misma categoría de software, pero difieren en algunos conceptos y valores [29].

La iniciativa OS adoptó las mismas pautas que el código abierto de Debian GNU/Linux [30]. De esta forma, los productos que son licencias OS, son productos que, además de dar libre acceso a su código fuente, satisfacen la Open Source Definition (OSD) [31]. Esta

definición establece formalmente cuales son las condiciones que debe cumplir un componentes para considerarse un componentes OS, y por tanto para que tenga un certificación como OSD. Las condiciones de la OSD son las diez siguientes [32]:

1. **Libre redistribución.** El software debe poder ser regalado o vendido libremente.
2. **Código fuente.** El código fuente debe estar incluido o poderse obtener libremente.
3. **Trabajos derivados:** La redistribución de modificaciones debe estar permitida.
4. **Integridad del código fuente del autor:** Las licencias pueden requerir que las modificaciones sean redistribuidas solo como parches.
5. **Sin discriminación de personas o grupos:** Nadie puede dejarse fuera.
6. **Sin discriminación de áreas de iniciativa:** Los usuarios comerciales no pueden ser excluidos.
7. **Distribución de la licencia:** Deben otorgarse los mismos derechos a todo el que reciba el programa.
8. **La licencia no debe ser específica de un producto:** El programa no puede licenciarse solo como parte de una distribución mayor.
9. **La licencia no debe restringir otro software:** La licencia no puede obligar a que algún otro software que sea construido con software de código abierto deba también ser de código abierto.
10. **La licencia debe ser tecnológicamente neutral:** Ninguna disposición de la licencia puede ser basada en una tecnología específica o en un estilo de interfaz. Es decir, no debe requerirse la aceptación de la licencia por medio de un clic del ratón o de otra forma específica del medio de soporte del software.

Los componentes OSS son componentes con licencia OSD. Los desarrollan proveedores comerciales o comunidades de código abierto. Al ser lanzados bajo las condiciones de la OSD, su código es abierto y disponible para su uso y cambio [33]. Siguiendo un modelo de desarrollo conocido como Bazaar [34].

Las principales ventajas de los componentes OSS son [32]:

- Menor costo
- Libre adquisición y utilización
- Modificabilidad
- Duplicabilidad
- Disponibilidad del código fuente
- Menor riesgo de soporte a los proveedores

Las principales desventajas de un componente OSS son:

- Necesidad de particularización para su uso en aplicaciones concretas.
- Falta de documentación y soporte.

### 2.1.1.2 Componentes Commercial-off-the-shelf (COTS)

También llamados componentes comerciales. Son propiedad de los proveedores comerciales, los clientes deben pagar por su uso, y no tienen acceso a su código fuente [35].

En la última década el uso de componentes COTS como parte de sistemas software basados en componentes ha crecido considerablemente. El uso de estos componentes afecta las actividades del proceso de desarrollo de software, lo cual ha llevado a investigar sobre este respecto.

Las definiciones de componente COTS encontradas en la literatura varían considerablemente. A continuación se enumeran algunas de las más referenciadas:

- De acuerdo con Obendorf [36], los componentes COTS son productos que pueden ser adquiridos, listos para ser usados en una estantería virtual de algún desarrollador (por ejemplo, a través de un catálogo o una lista de precios). Dichos componentes permiten obtener, a un costo razonable, algo que ya hace el trabajo. Que por otra parte reemplazan las dificultades de desarrollar componentes propios para un sistema único.
- Para Vidger y Dean [37], los componentes COTS son productos de software pre-existentes, vendidos en múltiples copias con cambios mínimos, cuyos clientes no tienen control sobre especificación, calendario, y evolución. Su código fuente así como documentación interna normalmente no están disponibles. Sus especificaciones de comportamiento completas y correctas tampoco lo están.
- Basili y Boehm [38] proponen otra definición de componente COTS que sigue las siguientes características: el comprador no tiene acceso a su código fuente y el proveedor controla su desarrollo y evolución.
- Carney y Long [39] consideran el origen y la modificación como atributos para definir los componentes COTS. Según su origen los componentes COTS son: componentes producidos por contrato, componentes existentes de fuentes externas, componentes producidos en casa. Según si pueden ser modificados o no, los componentes COTS requieren: re trabajo extensivo del código, revisión de código interna, adaptación y personalización necesaria, parametrización simple, y modificación escasa o ninguna.

Al revisar estas definiciones vemos que: el término componente COTS resulta genérico, cubriendo una gran variedad de productos. Todas las definiciones coinciden en considerar estos componentes como una clase especial de componentes reutilizables.

En las mismas fuentes de donde se han obtenido las definiciones, se clasifican los componentes COTS según los clientes a quien van dirigidos. Entre otros tipos distinguen: los componentes COTS en general, que serían componentes dirigidos al público en general; los componentes Military off-the-shelf (MOTS), que serían los dirigidos a un uso en aplicaciones militares; y los componentes Government off-the-shelf (GOTS), que serían los dirigidos a un uso en aplicaciones de la administración pública. También se usan términos

como: paquete de software, non development item (NDI), productos out-off-the-box, y solución shrink-wrap, como sinónimos de COTS [40].

En lo que se refiere a la presente tesis de máster, creemos que la definición mas adecuada seria la del Software Engineering Institute (SEI) [41]. Según esta definición un componente OTS es un producto de software que puede ser vendido, arrendado o licenciados al publico en general, ofrecido por un proveedor para obtener ganancias de él, soportado y evolucionado por el proveedor, quien tiene sus derechos de propiedad intelectual, disponible en múltiples e idénticas copias, y usado por el cliente sin modificación del código fuente.

Cabe aclarar que componente OTS es un término referente tanto a software como a hardware. Sin embargo en el ámbito de esta tesis de máster nos interesan únicamente los componentes de este tipo que son software.

### **2.1.2 Ingeniería de software basado en componentes**

Tradicionalmente, las organizaciones desarrollaban sistemas a partir de cero controlando así la totalidad o la mayoría de sus piezas, siguiendo los modelos establecidos en el proceso de ingeniería de software (modelos de cascada, espiral, o iterativos). Sea cual sea el modelo utilizado, se desarrollaban actividades como la elicitación de requisitos, el diseño, arquitectura, la construcción, la integración y las pruebas.

La necesidad de desarrollar aplicaciones software complejas en periodos de tiempo cada vez más cortos, así como la creciente oferta de componentes OTS en el mercado, ha hecho evolucionar el enfoque tradicional hacia un enfoque de desarrollo de software basado en componentes (CBSD). Este modelo de desarrollo permite construir una aplicación buscando y ensamblando componentes desarrollados por terceros, que combinados adecuadamente satisfacen los requisitos del sistema.

El concepto de construcción de software basada en subsistemas o componentes no es nuevo. Un diseño clásico de sistemas de software complejo comienza con la identificación de partes de sistemas diseñados con subsistemas o bloques y en menor escala módulos, clases, procedimientos, etc. Por lo que la ingeniería de software tradicional se ajusta a este nuevo enfoque, y la Component-Based Software Engineering (CBSE) se reconoce como una nueva sub-disciplina de la Ingeniería de Software.

Las metas de la CBSE son la reutilización, la adaptación y la extensión [42]:

- Soportar el desarrollo de sistemas construidos mediante componentes.
- El desarrollo de componentes como una entidad reutilizable. Un componente es reutilizable en la medida en que sus servicios pueden ser utilizados por otro software.
- El mantenimiento y mejoramiento de sistemas mediante la personalización y sustitución de componentes. Un componente es adaptable si su proveedor ha previsto los posibles cambios que puede sufrir dicho componente y es extensible si su proveedor proporciona los mecanismos para modificar los servicios que ofrece el componente.

Construir una aplicación se convierte por tanto en la búsqueda y ensamblaje de piezas prefabricadas, por lo que podemos decir que los Component-Based Systems (CBS) son sistemas contruidos por el ensamble de componentes OTS desarrollados con anterioridad y preparados para su integración [43]. Es importante aclarar que consideramos a CBS como una aplicación basada en la integración de uno o más componentes OTS, mientras que CBSD es el proceso que permite el desarrollo de un CBS.

El CBSD tiene varias ventajas que incluyen: Una gestión más eficaz de la complejidad, la reducción de tiempo de salida al mercado, el aumento de la productividad, la mejora de la calidad, un mayor grado de consistencia, y un amplio rango de usabilidad [44].

Sin embargo, existen varias desventajas y riesgos al usar CBSD [43], debido a que:

- Productos y tecnologías nuevas y mejoradas se ofrecen continuamente por los proveedores, lo cual hace que el mercado de los componentes COTS sea cada vez mas amplio
- Las actualizaciones y modificaciones de los productos están en manos de los proveedores y sus metas, y muy probablemente no corresponden a las metas de la organización en donde se implementa el componentes COTS
- Cada producto tiene su propio modelo para su uso y es probable que no corresponda con el modelo que sigue el usuario final
- Los proveedores lanzan nuevas versiones del software según sus necesidades y no las de la organización.
- Los procesos y la arquitectura utilizada para su construcción, pueden hacerlo difícil de integrar.
- La organización en ocasiones tiene que ajustar los procesos de trabajo para encajarlo al uso de componentes COTS

Por lo que se debe enfrentar a muchos desafíos desde problemas técnicos hasta legales con fin de adaptar las actividades de ingeniería de software tradicional a los beneficios de la utilización de componentes COTS [45, 46].

### **2.1.3 Características de la selección de componentes OTS**

Construir una aplicación mediante CBSD implica la búsqueda y ensamblaje de componentes OTS por lo que bajo este planteamiento, cobran especial interés los procesos de selección de los componentes apropiados para construir las aplicaciones.

Existen dos fases en la selección de componentes [10, 47]:

- Fase de búsqueda.
- Fase de evaluación.

**Fase de búsqueda.** En esta fase se identifican las propiedades de los componentes OTS, es decir, sus funcionalidades, los aspectos no funcionales como el uso de estándares y lo relativo a su calidad y aspectos no técnicos, como la cuota de mercado de sus vendedores o su grado de madurez.

**Fase de evaluación.** En esta fase se evalúan los candidatos de los componentes OTS encontrados en la fase de búsqueda con tal de seleccionar el más adecuado. Existen distintos métodos para efectuar el proceso de evaluación, Estos métodos toman en cuenta las necesidades de los dominios de aplicación y se basan en distintos modelos para conseguir la evaluación de los componentes OTS, tales como regresión, consensuales o de agregación.

El proceso de selección inicialmente descompone los requisitos para la selección de componentes OTS en un conjunto de criterios (incluidos en al fase de búsqueda). Este criterio incluye componentes, requisitos funcionales y no funcionales, restricciones de arquitectura y factores no técnicos como garantías del proveedor o temas legales. Entonces durante el proceso de selección, se identifican y analizan las propiedades de cada candidato OTS de acuerdo al conjunto de criterios de evaluación siguiendo una metodología.

A pesar de no existir un método comúnmente aceptado para la selección de componentes OTS [48] todos comparten pasos alternativos y superpuestos en cuanto a [49]:

- Búsqueda de componentes OTS en el mercado
- Evaluación de los candidatos de componentes OTS con respecto a los requisitos del sistema
- Decisión del mejor componente OTS a partir de la competencia de un conjunto de alternativas.

En la Tabla 2-A se muestran algunas de las funciones necesarias para realizar el proceso de selección de componentes OTS y se esboza la actividad de documentar la decisión [50]:

Actividad	Papel de usuario de componentes OTS
Busqueda de candidatos OTS	<i>Vigilante de mercado</i> explora el segmento de mercado para encontrar componentes que puedan concordar con los requisitos establecidos
Evaluación candidatos OTS	<i>Ingeniero de Calidad</i> mide los factores que están relacionados con los requisitos en los componentes candidatos.
Deducción de componentes OTS	<i>Seleccionador</i> toma la decisión final basado en la evaluación de los candidatos, tomando en cuenta otra información relevante (principalmente organizacional)
Documentación de la decisión	<i>Poseedor de conocimiento</i> almacena y documenta la información producida y las decisiones tomadas en el proceso para su uso en próximos procesos de selección.

**Tabla 2-A: Actividades los componentes OTS**

Es muy prometedor el uso de componentes OTS para mejorar la productividad y la calidad en el desarrollo de software. Sin embargo el uso de componentes OTS introduce nuevos problemas y riesgos, incluidos la dificultad de seleccionar componentes adecuados y el insuficiente análisis de requisitos [51]. Por un lado es conocido el hecho de que uno de los problemas esenciales en la reutilización de componentes de software es localizarlo y obtenerlo del vasto mercado de los componentes OTS [52] y por otra parte, la búsqueda de

componentes OTS hace frente a algunas características desafiantes [47] como las que se describen a continuación:

1. Crecimiento del mercado de componentes OTS: Debido a que productos y tecnologías nuevas y mejoradas se ofrecen continuamente, cada vez se ofrecen más productos y surgen nuevos segmentos de mercado; un claro ejemplo son las tecnologías móviles.
2. Cambios rápidos en el mercado de componentes OTS: Nuevas versiones de productos existentes son lanzadas cada poco tiempo y mas aun las fronteras de mercado se mueven con los años haciendo que los productos ofrezcan servicios que inicialmente pertenecían a otro segmento; un ejemplo los sistemas de servicio de correos usualmente proveían mensajes instantáneos y ahora ofrecen servicios de video conferencia.
3. Dependencias entre componentes OTS: Los componentes OTS no están diseñados para trabajar de forma aislada, sino con la colaboración de otros. Por lo que existen muchas dependencias entre ellos, tanto para permitir, mejorar o complementar su funcionalidad. Tal como los sistemas de manejo de documentos necesitan de herramientas de imagen de documentos para escanear y guardar los documentos.
4. Descripciones de componentes OTS disponibles en el mercado: los proveedores de componentes OTS no proveen un tipo estructurado de información que permita automatizar o al menos ayudar a la búsqueda, además no es realista pensar que esta situación vaya a cambiar en un futuro. Especialmente en los sistemas ERP, CRM y SCM. Esta situación se agrava por el hecho de que la información del proveedor tiende a mostrar las fortalezas y ocultar las debilidades de los componentes OTS.

Por lo que algunas condiciones generales para la selección de componentes COTS de acuerdo con [53] son:

- **Alcance del Dominio.** Los componentes COTS han de proporcionar las funcionalidades necesarias para satisfacer total o parcialmente los requisitos esenciales del cliente.
- **Restricción de tiempo.** Por lo general el calendario que sigue una empresa para un proyecto de software es muy rígido y de ahí depende su competitividad y la selección de componentes COTS es una actividad que consume un tiempo considerable necesario para la búsqueda y selección de todos los posibles candidatos.
- **Escala de costo.** El presupuesto disponible es una variable muy importante. Los gastos en el momento de seleccionar los componentes COTS serán influenciados por factores tales como: adquisición de licencia, gastos de soporte, los gastos de adaptación, y los costos de mantenimiento. En [54, 55] se proporciona un modelo económico para estimar el costo de componentes COTS sistema basado en el desarrollo.
- **Garantía del proveedor.** Un aspecto importante a considerar en la selección es verificar el soporte técnico proporcionado por el proveedor. Algunos de los aspectos que han de tenerse en cuenta, serían: la reputación del proveedor y su

madurez, el número y tipo de aplicaciones que han realizado utilizando componentes COTS, y las características de las cláusulas en las licencias de mantenimiento.

#### 2.1.4 Métodos de Selección de componentes OTS

Haciendo una revisión de la literatura, notamos la existencia de una variedad de métodos propuestos para la selección de componentes OTS. La mayoría proponen una forma para realizar la evaluación de los componentes candidatos y la forma en la que elegir el que más se ajusta a los requisitos de la organización.

A continuación se describen brevemente algunos de ellos:

**OTSO:** [56] (Off-the-Shelf-Option) Puede ser considerado el primer método extenso de selección; formula los principios básicos que serán incorporados por los métodos subsecuentes, enfatiza una sistemática y detallada definición de criterios de evaluación del reutilización de metas y utiliza técnicas formales como AHP. En principio se propuso para la selección de componentes COTS, pero una vez estudiado parece que se podría aplicar igualmente a cualquier componente OTS.

Este método comienza con la definición de criterios de evaluación jerárquicos que toman en cuenta varios factores de decisión, entre los que están que incluyen, la especificación de requisitos, la infraestructura organizacional, la arquitectura de la aplicación, los objetivos del proyecto y la disponibilidad de librerías. Todas las alternativas son comparadas con respecto al costo y al valor que proveen y el método de evaluación y comparación AHP es usado para consolidar la evaluación de resultados por decisión-making.

OTSO asume que los requisitos existen y que son fijos. Evalúa a los candidatos construyendo en primer lugar una jerarquía (ranking) de atributos a evaluar y después compara entre cada par posible en cada grupo (como una matriz), de acuerdo con la fortaleza de los atributos en las condiciones del proyecto, aplicando aquí el AHP.

**PORE:** [57] (Procurement-Oriented Requirements Engineering) es un método basado en un proceso iterativo de las necesidades de adquisición y evaluación de productos. Integra técnicas de ingeniería de requisitos, ingeniería de conocimientos, toma de decisiones multicriterio y análisis de características para guiar la selección de componentes OTS.

Propone un proceso concurrente, en el que las necesidades de adquisición de los stakeholder y la selección de componentes COTS se identifican al mismo tiempo para ayudar al equipo a tomar decisiones sobre qué componentes seleccionar y cuales rechazar.

Las diferencias de los componentes candidatos restantes son usadas para orientar los nuevos requisitos de adquisición, para lo cual hay que centrar el esfuerzo del equipo en la información más importante acerca de los requisitos y componentes de software para la toma de decisiones en cualquier momento.

PORE propone un modelo de cumplimiento product-requirement para tener una base teórica para la técnica de selección, la cual es hecha por rechazo.



**SCARLET:** [58] Formalmente conocido como BANKSEC es una extensión del método PORE, que lo adapta al dominio de la banca. Uno de sus principales objetivos es ayudar a crear sistemas que cumplan con los requisitos no funcionales usando componentes, para cumplir con este objetivo se integra PORE con modelos de requisitos no funcionales y componentes de especificación de lenguajes creando así procesos innovadores para ofrecer a los bancos europeos.

**CEP:** [59] Comparative Evaluation Process es una instancia del método DAR y esta basado en una estructura de proceso decision-making. Define los criterios de evaluación para el proceso. Los pesos se establecen para todos de los criterios de evaluación con respecto a la importancia de cada proyecto. La teoría de decisión detrás del modelo de evaluación es de simples promedios ponderados. Las ponderaciones son subjetivas y dependen de un proyecto en particular. Las categorías del criterio de evaluación incluyen metas básicas, de gestión, de arquitectura, estratégicas y funcionales y calcula la evaluación de resultados basado en el criterio de valor y las puntuaciones de credibilidad

**CARE:** [60] COTS-Aware Requirements Engineering es un método orientado a metas y objetivos. Sin embargo, el método no trata de utilizar la capacidad en los componentes OTS en el proceso de evaluación. En lugar de esto la tarea de evaluación de los candidatos OTS se confía con la ingeniería de requisitos. Este modelo define cuando y como definir los agentes, objetivos, requisitos y la arquitectura con respecto a la utilización de los componentes OTS.

Primeramente y mediante el uso de agentes, metas y arquitectura de software se propone la definición de procesos, definiendo metas, componentes relacionados, ranking de componentes, selección de componentes y negociación de cambios.

**CRE:** [53] COTS based on Requirements Engineering esta orientado a metas, se enfoca principalmente a la definición de los requisitos no funcionales para asistir a los procesos de evaluación y selección de componentes OTS. Basa la decisión de selección en la estimación de costo-beneficio y se hace por rechazo.

**PECA :** [61] Es llamado PECA ya que consiste de cuatro elementos: Planeación de la evaluación, establecimiento de criterios, Colectar datos, Analizar datos. La meta de PECA es proveer toda la información necesaria para que se tome una decisión. Se deriva del ISO 14598, el proceso es flexible.

**CAP:** [62] COTS Acquisition Process aborda el concepto de proceso de evaluación adaptable para componentes OTS destacando que el proceso de evaluación debe adaptarse basándose de en el esfuerzo disponible para cada proyecto. La adaptación del proceso se hizo en base al conocimiento de expertos.

**STACE :** [63] Socio-Technical Approach to COTS Evaluation, considera por complete el proceso de evaluación y selección de componentes COTS y enfatiza la participación del cliente durante la evaluación. Específicamente estudia como definir el criterio de evaluación; Existen tres principios principales, a) soporte para un enfoque sistemático en la evaluación y selección de COTS, b) soporte tanto para evaluación de componentes OTS como de tecnología subyacente y c) uso de técnicas socio-técnicas para mejorar el proceso de selección de componentes OTS. Este método soporta la negociación entre la elicitación de requisitos y la evaluación de componentes OTS.

**Story Board:** [64] Sugiere incorporar casos de uso y capturas de pantalla durante el proceso de requisitos para ayudar a los clientes a entender sus requisitos y así, adquirir el componente COTS mas apropiado.

Por otra parte, hay algunas técnicas de evaluación relativamente maduras para seleccionar de entre un grupo de productos por pares. Por ejemplo, la Organización Internacional de Normalización (ISO) [65] describe los criterios generales para la evaluación de productos mientras que otros describen las técnicas que toman en cuenta las necesidades particulares del dominio de aplicación [66, 67]. Estos criterios de evaluación suele implicar una combinación de los estudios de los componentes COTS, discusiones con otros usuarios de estos componentes COTS, y la evaluación comparativa y prototipos.

### **2.1.5 Selección de componentes en el grupo GESSI**

Una de las áreas de investigación del grupo GESSI es la selección de componentes OTS, y más concretamente se ha trabajado en el análisis y la mejora en los procesos de selección de componentes OTS. A continuación se presentan las principales aportaciones realizadas por el grupo en este ámbito.

Uno de los temas en el que se han hecho aportaciones es en el proceso de búsqueda de candidatos de componentes COTS [68] proponiendo tanto el uso de algunas técnicas apoyando tanto a la comprensión y confiabilidad de la estructura de los componentes, como la reutilización de componentes COTS.

Otro tema en el que se han hecho aportaciones es en el alineamiento entre requisitos y características de un componente COTS [69]. Concretamente se ha propuesto una guía usando modelos de calidad para potenciar los objetivos y las características de los componentes COTS.

Por otra parte, el grupo también ha trabajado en uno de los problemas actuales de la selección de componentes OTS, concretamente en el problema de cómo hacer frente a la gran cantidad de información no estructurada, incompleta y generalizada que existe sobre los dominios y productos software, y que aumenta el riesgo de tomar decisiones equivocadas durante los procesos de selección.

Con el fin de tener a mano información crucial pertinente a los diferentes componentes OTS [70] se basa en la idea de colaboración open source para la infraestructura que permita la reutilización de componentes OTS, permitiendo apoyo automático a los procesos de selección.

Centrarse en los factores más importantes que influyen en la selección es tarea básica para seleccionar adecuadamente. Por lo cual, [71] propone diferentes dimensiones de interés para la selección de componentes COTS que son cubiertos por los diferentes modelos de dominio. En [72] se propone que la selección sea vista como un proceso de alineamiento entre el modelo organizacional y modelos de componentes COTS. En [73] se estudian los principios ágiles en el contexto de selección de componentes COTS

Se realiza un estudio de las metodologías de selección de componentes COTS existentes [74]. Se proponen herramientas y métodos para llevar a cabo la selección, tal es el caso del El método GOTHIC [75] que provee una metodología para la construcción de taxonomías para la localización, análisis y estructuración de componentes COTS. [76] presenta un

enfoque para gestionar requisitos no técnicos durante la selección de componentes COTS extendiendo el modelo de calidad ISO/IEC 9126-1 [77]. DesCOTS [78] un sistema para ayudar a los clientes en la selección de componentes COTS. Se basa en el uso de modelos de calidad para evaluar los componentes COTS de un determinado dominio. DesCOTS-SL [79] apoya la definición de las necesidades. DesCOTS-DV [80] ayuda en la evaluación de componentes COTS.

Los modelos de calidad se consideran adecuados para la selección de componentes COTS debido a sus características. Por ello en [81] se presenta un resumen de las experiencias de GESSI en el uso de modelos de calidad para la selección de componentes COTS.

A partir de este caso de estudio [82] se demuestra la explotabilidad de REACT demostrado en el mundo real de componentes COTS y en el ejercicio de selección de componentes COTS.

## 2.2 Ingeniería de Requisitos

La definición de requisito propuesta en [83] y en el estándar IEEE 610.12-90 define un requisito como: Una condición o capacidad que una aplicación debe hacer o tener para resolver un problema o alcanzar un objetivo. Un requisito existe tanto si la aplicación exige ciertas cualidades o funciones o el cliente desea que esos requisitos sean parte de la aplicación.

La ingeniería de requisitos constituye una fase temprana del ciclo de vida del desarrollo de software que corresponde a la especificación de requisitos, es decir, a la especificación de lo *qué* debe ser implementado [1]. Esto implica la comprensión del dominio de la aplicación, las restricciones, la funcionalidad específica requerida por las partes interesadas o stakeholders (personas que directa o indirectamente utilizarán el sistema o la información que proporciona) y otras características esenciales del sistema como rendimiento y seguridad.

Es una realidad que los requisitos causan dificultades [1]. Algunas de las manifestaciones visibles de estas dificultades son:

- Las necesidades reales pueden no estar cubiertas debido a malentendidos en la interpretación de los requisitos.
- Los requisitos pueden estar incompletos.
- Los requisitos pueden variar debido a cambios en el entorno.
- La utilización del lenguaje natural puede provocar la ambigüedad de los requisitos.
- Puede no existir una técnica adecuada para articular todas las necesidades del sistema.
- Pueden faltar herramientas apropiadas que ayuden a la ingeniería de requisitos a estructurar y formular eficientemente los requisitos.

La ingeniería de requisitos debe proveer e incluir actividades para tratar de solucionar estas dificultades. La ingeniería de requisitos es el nombre que corresponde a un conjunto estructurado de actividades que ayudan a desarrollar la comprensión y la documentación de los requisitos.

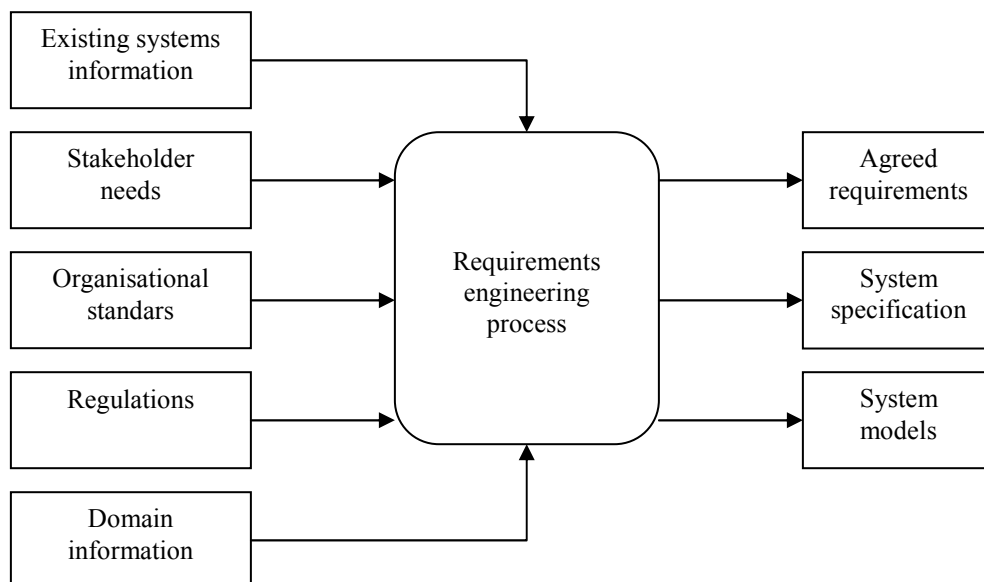
En este capítulo analizaremos las siguientes secciones. En la sección 2.2.1 se presenta la definición de ingeniería de requisitos y sus principales características. En la sección 2.2.2 se describe el ciclo de vida de la ingeniería de requisitos. En la sección 2.2.3 se presentan los tipos de requisitos, sus características y algunos ejemplos. En la sección 2.2.4 se describe la elicitación de requisitos y sus características. En la sección 2.2.5 se define el concepto de reutilización y las características de la reutilización. Finalmente, en la sección 2.2.6 se presentan las aportaciones del grupo GESSI en requisitos hasta el día de hoy.

## 2.2.1 Definición de Ingeniería de Requisitos

Según [84] la ingeniería de requisitos (RE) es el área de conocimiento de la ingeniería de software relativa a la obtención de objetivos, funciones y restricciones de los sistemas de software en el mundo real. Además se encarga de la relación de estos factores para precisar las especificaciones del comportamiento del software, y su evolución a lo largo del tiempo y a través de familias de software.

En términos generales, la ingeniería de requisitos es el proceso de descubrir el objetivo del sistema de información, mediante la identificación de los interesados (stakeholders) y sus necesidades.

El objetivo principal del proceso de ingeniería de requisitos es proporcionar un modelo de las necesidades del problema que debe resolverse de forma clara, coherente, precisa y no ambigua y debe incluir el medio ambiente con el que interactúa. Este modelo se debe documentar de forma que pueda ser susceptible al análisis, comunicación, y su aplicación posterior. El proceso de ingeniería de requisitos es un proceso de diseño con entradas y salidas (ver figura 2-A).



**Figura 2-A: Entradas y salidas del proceso de ingeniería de requisitos**

### 2.2.2 El ciclo de vida de la ingeniería de requisitos

El ciclo de vida de la ingeniería de requisitos consiste en (ver figura 2-B) [85]:

- **Elicitación:** Identificación de las fuentes de información relacionados con el sistema y descubrir los requisitos de este.  
**Análisis:** Estudio de los requisitos obtenidos, sus superposiciones y conflictos.
- **Validación:** Comprobación de que los requisitos obtenidos corresponden a lo que realmente se necesita por parte de los stakeholders (los interesados).
- **Negociación:** Estimación de las necesidades pudieran entrar en conflicto; conciliación de los puntos de vista de los diferentes stakeholders; y generación de un conjunto coherente de necesidades.
- **Documentación:** Redacción de los requisitos de forma que los stakeholders y los desarrolladores de software puedan entender.
- **Gestión:** Control de los cambios en los requisitos, ya que inevitablemente estos cambios se producirán.

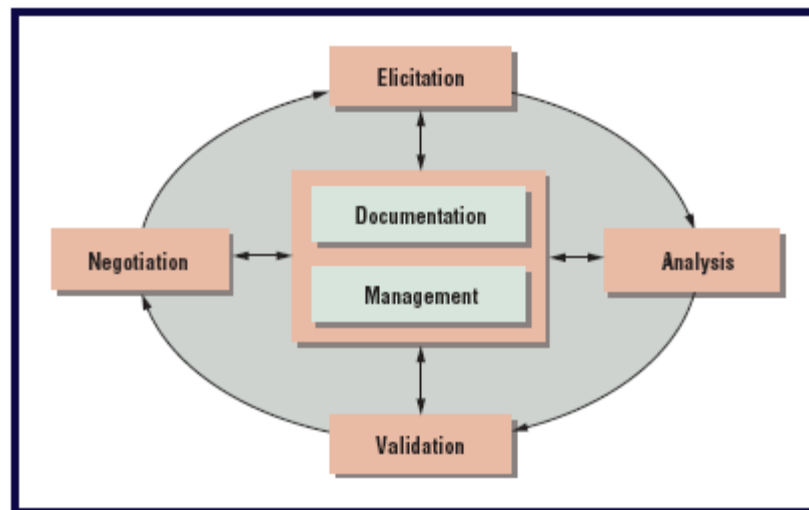


Figura 2-B: Ciclo de vida de la ingeniería de requisitos

### 2.2.3 Tipos de Requisitos

Los requisitos se pueden clasificar en funcionales, no funcionales o no-técnicos. A continuación se describen los tres tipos de requisitos.

#### 2.2.3.1 Requisitos Funcionales

Los requisitos funcionales según [83] especifican funcionalidades que un software debe proveer a sus usuarios. Una funcionalidad es lo que debe hacer un producto de software, se refiere a una acción que el producto debe realizar. Los requisitos funcionales son la razón fundamental de la existencia de producto, por consiguiente:

Un requisito funcional es [83]:

- La especificación de la funcionalidad de un producto.

- Una acción que un producto debe proveer (chechar, calcular, grabar, devolver, etc.).
- Un derivado del propósito fundamental del producto.
- No debe ser una necesidad que exprese la calidad con la que se provee una determinada función o acción (rápido es un requisito de calidad, pero no un requisito funcional).

Algunos tipos de requisitos funcionales:

- Requisitos sobre la actualización de datos: características sobre las funciones que cambian la información del sistema. Debe estudiarse de qué forma y bajo qué restricciones el usuario desea que se introduzcan nuevos datos, se cambien los que ya existen o se eliminen.
- Requisitos sobre la estructura de la información: características de los datos que el software maneja.
- Requisitos sobre las consultas y los informes.
- Requisitos sobre la interacción con otros sistemas.

Ejemplos de requisitos funcionales:

1. Se deben poder realizar búsquedas en base a diferentes criterios.
2. Cada factura tendrá un número único y correlativo.

### **2.2.3.2 Requisitos No Funcionales**

Los requisitos no funcionales según [83] son cualidades o propiedades que un producto debe tener. Generalmente están relacionados con una funcionalidad. Esto es, cuando se sabe lo que un producto debe hacer, se puede determinar como lo debe hacer, que cualidades debe tener y que tan grande y rápido debe de ser.

Algunos tipos de requisitos no funcionales:

- Requisitos de rendimiento: Límites al rendimiento (para aquellas aplicaciones donde existan) y volúmenes de información que el software debe tratar.
- Requisitos de frecuencia de tratamiento: Características sobre la frecuencia con que se ejecutan las diferentes funciones del software.
- Requisitos de seguridad: Características de control de acceso al software y copias de seguridad, entre otros relacionados con la seguridad del sistema y la información.

Ejemplos de requisitos no funcionales:

1. El producto debe usar los colores de la compañía, los logos y los tipos de letra estándar.
2. Se utilizará en todas las comunicaciones el conjunto de caracteres ADA estándar.

### **2.2.3.3 Requisitos No Técnicos**

Los requisitos no técnicos según [76] se refieren a características externas de los componentes, en el contexto de su comercialización, tales como características de la organización que los comercializa, características de licencia, etc.

Algunos tipos de requisitos no técnicos:

- Proveedor: Aspectos relacionados con proveedores que pueden afectar la calidad del componente (fortaleza, fiabilidad, servicios, soporte, etc.).
- Adquisición: Evalúan aspectos que usualmente preocupan a las organizaciones al momento de adquirir un componente (garantías, propiedad, licenciamiento, etc.).
- Producto: aspectos relacionados con el historial del producto y la facilidad para su implementación (historia, entregables, parametrización y personalización, etc.).

Ejemplos de requisitos no técnicos:

1. El producto debe tener un distribuidor local.
2. El producto debe tener un servicio técnico en línea que atienda cualquier incidencia en menos de 1 hora.

## **2.2.4 Elicitación de Requisitos**

El punto crucial en un proyecto de software es la determinación de los requisitos que este debe satisfacer. La correcta especificación de los requisitos del software se asocia con la satisfacción de las necesidades y expectativas del cliente [86-88]. Por lo que se reconoce que un factores clave para el buen funcionamiento del software, es el acoplamiento de los objetivos de la organización a los requisitos del software.

La elicitación de requisitos es el nombre común que se le da a las actividades correspondientes a determinar los requisitos del sistema. Consiste en un análisis completo de la organización, el dominio de aplicación y los procesos de negocio donde se utilizara el sistema [1].

La elicitación de requisitos es un proceso complejo, y uno de los más cruciales en el ciclo de vida del software. La elicitación de requisitos consiste en capturar las necesidades reales que se tienen que ver reflejadas en el sistema. Para el cual existen 4 dimensiones [1]:

1. Entendimiento del dominio de aplicación: Es el conocimiento general del área donde se aplicara el sistema.
2. Entendimiento del problema: Consiste en comprender el detalle de las especificaciones del problema que el cliente quiere resolver.
3. Entendimiento de la organización: Se debe entender como el sistema interactúa y afecta las diferentes partes de la organización y como puede contribuir a alcanzar los objetivos de la organización.
4. Entendimiento de las necesidades y restricciones de los stakeholders: Los Stakeholders son todas aquellas personas a quienes el sistema afecta de alguna forma. Pueden ser usuarios finales, administrativos, etc. Se debe entender las necesidades específicas de su área de trabajo y de su rol.

La elicitación no es una tarea fácil y se necesita refinar los requisitos constantemente. Por lo que la elicitación va ligada al análisis y la negociación. Debido a que los requisitos son analizados inevitablemente una vez que han sido descubiertos y en muchos casos se negocia para llegar a un acuerdo en la definición de los requisitos antes de ser incluidos en el documento o libro de requisitos. Por lo que se puede pensar en estos tres procesos como una espiral en donde existe un proceso iterativo entre estos procesos (ver figura 2-C).

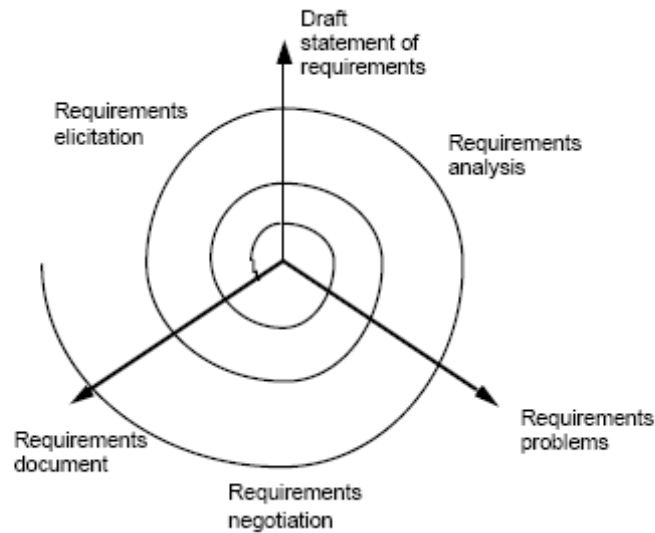


Figura 2-C: Espiral de elicitación análisis y negociación

Existen varias técnicas para facilitar el proceso de elicitación, tales como obtención de documentos, entrevistas, observación y análisis social, Joint-Application-Development (JAD), construcción de prototipos, casos de uso, storyboards, etc.

Del resultado de la aplicación de estas técnicas resulta entonces el conjunto de requisitos necesarios para cumplir los objetivos de la organización. Estos requisitos son almacenados y puestos en una forma escrita, denominada *libro de requisitos*.

El libro de requisitos posee entre otras cosas la especificación de los requisitos funcionales, no funcionales y no técnicos, que puede servir como base (además de la construcción de un sistema de software) para la construcción de contratos, call for tenders, etc.

### 2.2.5 Reutilización

La primera vez que se habló formalmente de reutilización en el ámbito de ingeniería de software fue en 1968, cuando en [89] se propuso la creación de fábricas de elementos de software análogas a las ya existentes de componentes hardware. Esta propuesta ha evolucionado aunque todavía queda mucho por hacer en este ámbito.

La reutilización responde al principio de aprovechar esfuerzos previos y exitosos para completar un nuevo desarrollo. La reutilización se considera clave para mejorar la calidad y la productividad a nivel de desarrollo de software.

Tradicionalmente se han diferenciado dos metodologías para enfocar la reutilización de software; la primera se basa en la obtención de un nuevo sistema por composición de elementos ya existentes y la segunda en la generación del nuevo sistema utilizando como base una estructura o modelo [90].

- **Reutilización por composición:** Es bastante intuitiva, se trata de conjugar elementos o componentes para construir un sistema nuevo. Aunque conceptualmente es sencilla, requiere un soporte técnico complejo como métodos de clasificación y selección optimizados según los componentes, y soporte para su adaptación e integración en el nuevo sistema.



- **Reutilización por generación:** Es conceptualmente más compleja, ya que no es posible definir los componentes como entes auto-contenidos y concretos. En este caso se reutilizan procesos de generación obtenidos como resultado de una *codificación de estructuras*. Se suelen distinguir tres subtipos de metodologías: generadores de aplicación, generadores basados en el lenguaje y sistemas transformacionales.

En las metodologías existentes para reutilización, se proporciona una sintaxis clara y una semántica definida para la representación de requisitos [21], sin embargo, es un tema del que hay poca evidencia en la literatura como para sugerir que se practica extensamente [5] sobre todo en las fases de elicitación [21].

### 2.2.5.1 Definición de Reutilización

Se puede decir que, la definición de reutilización más ampliamente aceptada corresponde a [91] que la define como:

"Reutilización de software es el proceso de creación de sistemas de software a partir de software existente en lugar de construirlos a partir de cero"

Debido a la flexibilidad de la definición, utilizaremos esta y nos referiremos a la *reutilización* como a la aplicación de artefactos de software ya existentes para la creación de un sistema nuevo, es decir, la reutilización de "*cualquier tipo de información ya existente que necesite el desarrollador de sistemas para la elaboración de un nuevo sistema software*"[92].

### 2.2.5.2 Reutilización de requisitos

La reutilización abarca diferentes productos del ciclo de vida dentro de los diferentes niveles de abstracción, incluyendo productos obtenidos en la fase de requisitos. Y es un hecho aceptado que la obtención de los mayores beneficios en la reutilización se debe a la comprensión precoz de la funcionalidad del sistema.

La reutilización de requisitos es un área prometedora para la ingeniería de requisitos [93]. Diferentes autores [94-98] recomiendan el abordar la reutilización lo más temprano posible en el ciclo de vida del software.

Es generalmente bueno reutilizar el mayor conocimiento posible cuando se desarrolla un nuevo sistema. La posibilidad de reutilizar diseños y código es fácil de entender, pero la reutilización de requisitos es un poco más complicada (debido a que se piensa que los requisitos de cada sistema son distintos debido a que el sistema es conceptualmente distinto y a que se tienen diferentes stakeholders y por tanto diferentes puntos de vista). Sin embargo la reutilización de requisitos es posible [1].

Los requisitos se pueden reutilizar en distintos enfoques como puede ser [99]:

- En desarrollo de nuevas especificaciones.
- En mantenimiento de las aplicaciones existentes.
- En desarrollo de nuevas aplicaciones.
- En selección de OTS.

Independientemente del enfoque que se le de, la reutilización de requisitos es análoga a la reutilización del software por lo que requiere formas de representar, almacenar, comparar y adaptar los elementos potencialmente reutilizables [21].

Con el fin de mejorar la calidad de los requisitos reutilizables, mejorar la productividad de los analistas de requisitos, entre otros, se han utilizado patrones ya que facilitan la expresión lógica de los requisitos [100].

Además la reutilización mediante el uso de patrones constituye una solución eficiente por distintas razones:

- La orientación del problema/solución motiva a los diseñadores a la búsqueda de soluciones a problemas dados.
- El tamaño reducido del patrón mejora su utilidad, comprensión, selección y adaptación.
- El alto grado de generalización de un patrón amplía su campo de aplicación.

La reutilización mediante el uso de patrones y los avances que existen en este ámbito se analizarán en el siguiente apartado del estado del arte.

## **2.2.6 Requisitos en el grupo GESSI**

Las aportaciones en modelos basados en objetivos para modelar sistemas orientados a servicios son necesarias para hacer frente a las necesidades heterogéneas de los stakeholders y la evolución de estos sistemas. Por lo que en [101] se propone complementar el modelo  $i^*$  con modelos de decisión mediante modelado de variabilidad ortogonal.

En [76] se propone que la selección de componentes OTS se haga no sólo de un análisis de su calidad técnica, sino también considerando cómo cumplir con los requisitos no técnicos que son relevantes para la selección de componentes OTS, agregándolos al modelo de calidad ISO/IEC 9126-1.

En [102] se describe la importancia de los requisitos en la selección de componentes OTS y su papel como conductores para personalizar un componente OTS. Se presenta además la forma de integrar las actividades clásicas de ingeniería de requisitos con sistemas basados en OTS.

En [103] se expone la idea de cómo las aportaciones en negotiation-based pueden ayudar a hacer frente a los desafíos planteados en ingeniería de requisitos mediante open MSDS y se presenta una plantilla para explicar la relación de los distintos aspectos de MSDS, así como su investigación sobre la conciliación de requisitos y arquitecturas.

El estudio presentado en [104] se centra en la posibilidad de desarrollar mecanismos para capturar requisitos no técnicos. En particular, se centra en la posible ampliación del Lenguaje Unificado de Modelado (UML) para obtener una especificación inicial de algunos requisitos no funcionales.

## 2.3 Patrones en Ingeniería de Software

Un patrón [105] corresponde a una solución de un problema en un contexto. Ayuda a crear un lenguaje común para comunicar ideas y experiencias acerca del problema y su solución [106]. La idea de patrón se basa en las siguientes observaciones:

- Algunos problemas son recurrentes.
- Las buenas soluciones que han ayudado a resolver dichos problemas con éxito deben ser compartidas.

Los patrones han sido usados en distintos ámbitos de la ingeniería del software, creados en primera instancia para resolver problemas identificados durante el diseño de paquetes software [10]. En los últimos años se ha empezado a hablar sobre el uso de patrones para ayudar en la ingeniería de requisitos, y en algunos casos más en concreto en la elicitación de requisitos[5].

A continuación exploraremos el estado del arte de los patrones en ingeniería de software. En la sección 2.3.1 se describe la historia del surgimiento de los patrones. En la sección 2.3.2 se expone la definición de patrón y sus principales características. En la sección 2.3.3 se muestra el surgimiento de los patrones en la ingeniería del software y los tipos más comunes. En la sección 2.3.4 se describe la utilización de patrones en la ingeniería de requisitos.

### 2.3.1 Historia

El nacimiento del término patrón se debe al arquitecto Christopher Alexander en 1979, quien en su libro *The Timeless Way of Building* proponía el uso de patrones para aumentar la calidad en la construcción de edificios.

Alexander quería crear estructuras que fueran buenas para las personas y, tener una influencia positiva en ellas mediante la mejora de su comodidad y su calidad de vida. Con este objetivo llegó a la conclusión de que los arquitectos deben luchar constantemente para producir productos que se ajusten y se adapten mejor a las necesidades de todos sus habitantes. Para ello, describió sus ideas para alcanzar estas metas, creando así los patrones de arquitectura. Para él: "Cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez."

Posteriormente, en 1987, Ward Cunningham y Kent Beck decidieron utilizar algunas de las ideas de Alexander para desarrollar un pequeño lenguaje de cinco patrones para guiar a los programadores principiantes de la herramienta Smalltalk. Publicando los resultados en [107].

Poco después, Jim Coplien comenzó a compilar un catálogo de modismos C++ (que son una especie de patrón) y en 1991 lo publicó como libro [108].

De 1990 a 1992, varios miembros de la llamada *Gang of four* se habían conocido y habían hecho algún trabajo de compilar un catálogo de patrones. Los debates sobre los patrones abundaban en un workshop Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'91) dado por Bruce Andersen (que se repitió en 1992).

En agosto de 1993, Kent Beck y Grady Booch patrocinaron un retiro en las montañas de Colorado, la primera reunión de lo que ahora es conocido como Hillside Group. Otro workshop de patrones se celebró en OOPSLA'93 y luego en abril de 1994, el Hillside group se reunió de nuevo (esta vez con Richard Gabriel) para planificar la primera conferencia Pattern Languages of Programs (PLOP).

Hoy en día los patrones se utilizan como referencias para describir soluciones a problemas de ingeniería y sirven de guía a los procesos de software [109].

### 2.3.2 Definición y características

Según [110] un patrón de diseño es una solución a un problema que se usa repetidamente en contextos similares con algunas variantes en la implementación. La forma de obtener esta solución es a partir de la abstracción de ejemplos específicos de diseño. Para que una solución pueda ser considerada un patrón de diseño debe ser *eficaz* (que se haya demostrado resuelve satisfactoriamente el problema) y *reutilizable* que pueda ser aplicada en diferentes casos.

En [111] se define el término patrón como la abstracción de una forma concreta que se mantiene recurrente en un contexto específico no arbitrario.

Según [112] cada patrón es una regla de tres partes, que expresa una relación entre un determinado contexto, un cierto sistema de fuerzas que ocurre repetidas veces en ese contexto, y una cierta configuración de software que permite a esas fuerzas resolverse a si mismos.

Por su parte, en [113] se da una definición de patrones para prendas de vestir diciendo: “Yo podría decirle cómo hacer un vestido especificando la ruta de una tijera a través de un trozo de tela en términos de ángulos y longitudes de corte. O bien, puedo darle una pauta. Al leer el pliego de condiciones, usted no tiene idea de lo que se está construyendo o si ha construido lo correcto cuando haya terminado. El patrón prefigura el producto: es la regla para hacer la cosa, pero es también, en muchos aspectos, la cosa en sí”.

Por lo que de forma general podemos decir que: Un patrón es la generalización o abstracción de una solución a un problema concreto que se repite en un contexto y proporciona tanto la forma de hacerlo (las reglas) como el artefacto en si.

Según [110] un patrón tiene las siguientes características:

**Encapsulación:** Cada patrón encapsula un problema bien definido y su solución. Los patrones son independientes, específicos, y su redacción debe ser suficientemente clara para precisar cuándo se pueden aplicar y para asegurarse que cada instancia o uso del patrón de como resultado la construcción de una entidad reconocible que forme parte del sistema en que participa.

**Generalidad:** Cada patrón debe contener una descripción de cómo aplicarlo, los patrones deben estar escritos para que los puedan utilizar todos los participantes, en la construcción del sistema, no solamente los encargados expertos de alguna manera, muchos patrones son como recetas que se ajustan cada vez que un experto guía su uso.

**Equilibrio:** Cada patrón debe identificar el espacio de su solución,(de su aplicación), y debe contener las constantes y las restricciones que minimicen los posibles problemas que

pueda generar en el medio ambiente dónde se inserta. Cuando un patrón se usa en una aplicación, El equilibrio es una parte estructural del patrón, Alexander lo reconoce como “la cualidad sin nombre” ya que es difícil de alcanzar en la mayoría de los patrones.

**Abstracción:** Los patrones representan abstracciones de evidencia empírica, son generales en ciertos contextos, pero no son necesariamente universales. La construcción de los patrones es un proceso social, iterativo, de coleccionar, compartir y amplificar experiencia y conocimiento. Los patrones también pueden surgir de partes de sistemas ya construidos, algunas veces de análisis de sistemas, de sus partes y relaciones con otros sistemas. La abstracción necesaria para la formación de patrones surge con las heurísticas basadas en análisis y diseño de equipo, introspección sobre conjuntos de requisitos, en general la abstracción de partes de sistemas existentes a patrones, a demás del consenso social de los involucrados aumenta la probabilidad de consolidación de un patrón.

**Apertura:** Los patrones se utilizan al buscar en ellos las referencias de las características deseadas en el proyecto que se trate, a estas características se puede llegar con otros patrones o con una combinación de ellos, lo que lleva a diferentes niveles de detalle y abstracción. Los patrones deben estar construidos con la apertura suficiente para organizarse de diversas maneras, sin perder su esencia ni soporte teórico.

**Agregabilidad:** Los patrones forman una jerarquía, una taxonomía, los de grano grueso se encuentran en las partes superiores de la clasificación y en las inferiores los patrones más finos, de más detalle. La mayoría de los patrones son agregables hacia arriba u hacia abajo y hay que dejar claras estas interacciones.

### 2.3.3 Patrones en ingeniería de software

Los patrones en ingeniería del software se volvieron populares con la aceptación del libro Design Patterns: Elements of Reusable Object-Oriented Software [10], en donde se define los patrones de diseño. Debido a esto, gran parte de los patrones iniciales en la comunidad del software se basan en los patrones de diseño.

La definición de patrones de diseño en [10] se centra en concreto en patrones de diseño orientado a objetos, sin embargo con pequeños cambios puede ser ajustada para describir patrones de diseño de software en general, teniendo que:

- Un patrón de diseño nombra, resume, e identifica los aspectos clave de una estructura común de diseño que lo hace especialmente útil para la creación de diseño orientado a objetos reutilizable.
- El patrón de diseño identifica las clases participantes y sus instancias, sus funciones y colaboraciones, y la distribución de responsabilidades.
- Cada patrón de diseño se centra en un problema particular de diseño orientado a objetos.
- En un patrón se describe cuando puede o no aplicarse en vista de unas determinadas restricciones de diseño, así como las consecuencias y los compromisos de su uso.
- Habitualmente da ejemplos de su aplicación. Por ejemplo, el código que ilustra su aplicación. A pesar de describir los patrones de diseño orientado a objetos, se basan en soluciones prácticas que se han implementado en los principales lenguajes de programación orientada a objetos.

Existen muchos otros tipos de patrones de software, además de los patrones de diseño, como los usados para la modelización de la organización, la planificación de proyectos, la ingeniería de requisitos, y la configuración del software de gestión (por nombrar algunos). En la actualidad, los más usados en la comunidad del software son los patrones arquitectónicos, los patrones de diseño de software, y (más recientemente) los patrones organizacionales:

**Patrones arquitectónicos:** Un patrón arquitectónico expresa una estructura de la organización o esquema de sistemas de software. Proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades, e incluye las normas y directrices para la organización de las relaciones entre ellos.

**Patrones de diseño:** Un patrón de diseño establece un posible refinamiento de los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. En él se describe comúnmente la estructura de componentes que pueden resolver un problema recurrente de diseño general dentro de un contexto particular.

**Patrones organizacionales:** Son patrones que se pueden situar en el nivel más abstracto de la ingeniería de requisitos. Un patrón organizacional presenta una propuesta de modelado de una organización. Estos patrones se basan en la teoría de organizaciones.

Muchas veces se habla de patrones de diseño refiriéndose a cualquier tipo de patrón.

### 2.3.4 Patrones en ingeniería de requisitos

En un conjunto de requisitos, uno o más se pueden repetir. Se pueden encontrar condiciones similares, ya sea por su expresión (petición o afirmación) o por sus características. Los patrones de requisitos ayudan a realizar descripciones genéricas sobre características específicas de los mismos.

En ese sentido los patrones de requisitos son generalizaciones de afirmaciones o peticiones.

Para llevar a cabo esta generalización, se sigue un proceso de abstracción y clasificación que nos conduce a la construcción de un patrón.

Los patrones, como soluciones generales, contienen los aspectos esenciales de los requisitos, tratan de captar las propiedades fijas de los requisitos, las propiedades comunes de los requisitos repetidos una y otra vez en los proyectos, así, en su formación existen algunos con mucho mas repetición e insistencia en ellos que otros, en algunos, sus propiedades tienen más evidencia empírica que otros, por ende, en su aplicación, unos son más auténticos y profundos que otros.

El uso de patrones como ideas bien documentadas permite evitar ambigüedades o malinterpretaciones causadas por el lenguaje natural en el que normalmente están expresados los requisitos. Por lo que los patrones ayudan a mejorar la expresión, la comunicación y la comprensión de los requisitos. Permitiendo validar los requisitos con el usuario final.

Los requisitos pertenecen a dos niveles de documentos [114]:

- La especificación de requisitos de usuario: Describen el conjunto de requisitos del cliente
- La especificación de requisitos de sistema: Describen el conjunto de requisitos técnicos necesarios para proveer estos servicios.

El número de propuestas de reutilización en ingeniería de requisitos es aún escaso, sobre todo a nivel de requisitos de cliente.

### **2.3.5 Propuestas existentes de uso de patrones de requisitos**

En la Tabla 2-B se muestran algunas de las diferentes propuestas existentes en la literatura para la aplicación de patrones en ingeniería de requisitos. De estas propuestas podemos observar que son pocas las propuestas en las que los patrones son utilizados para la creación de los documentos generados de la elicitación. Aquellas en las que los patrones se generan con este fin, no muestran claramente los pasos a seguir para la obtención de patrones de requisitos, ni la forma en la que deben ser escritos para que a pesar de estar en lenguaje natural, sean formales y entendidos tanto por clientes como por técnicos (desarrolladores e ingenieros de requisitos)

Criterio	Requirement Engineering Patterns Repository (REPARE) [123, 124]	Sharing Requirements Engineering Experience Using Patterns [109]	Extensible Requirement Patterns of Web Application for Efficient Web Application Development (XRPWeb) [122]	Requirement Patterns for Embedded Systems [121]	Requirement Patterns [120]	A Pattern Based Approach for Requirements Engineering [118, 119]	Identificación de Patrones de Reutilización de Requisitos en Sistemas de Información [105]	A variant Approach to Product Definition by Recognizing Functional Requirement Patterns [115]
Uso	Método general para capturar e intercambiar mejores practicas de RE de éxito	Guía para acceder fácilmente a herramientas y métodos probados de ingeniería de requisitos	Aplicaciones web	Sistemas integrados	Documentación de requisitos para separar sus especificaciones	Asistencia en elicitación de requisitos y como guía en el análisis de la organización	Reutilización vertical a distintos niveles de abstracción del desarrollo de software	Definición y diseño del producto
Resuelve	Actividades conflictivas en ingeniería del software	Inexistibilidad a los procesos definidos, acordados y establecidos dentro de una organización a nivel de gestión	Tiempo de salida al mercado, redundancia en el trabajo de aplicaciones web.	Uniformidad y fácil entendimiento de las estructura de creación de sistemas, rápida construcción de modelos de requisitos para sistemas	Velocidad en el proceso de especificación y captura de las necesidades sin ayuda de un ingeniero de requisitos	Soporte en la comunicación con el usuario final, justificación de elección de requisitos.	La complejidad del desarrollo del software y la especificación de requisitos	Definición de necesidades y riesgos en las especificaciones de requisitos para lograr el éxito del producto en el mercado
Base	Uso de patrones vectoriales relacionado a una tarea (2 fuerzas y 1 acción)	“The windows place” propuesto por Alexander para eliminar las situaciones con estrés	Reconstrucción y reutilización de patrones de requisitos de forma extensible	Lenguaje UML para representar patrones de requisitos comunes			Clasificación de utilización descrita en [114, 117] y casos de uso para representar los patrones	Una técnica de maquina de aprendizaje [116]

**Tabla 2-B: Algunas propuestas existentes sobre el uso de patrones**



## 2.4 Calidad de Software

Existe una gran cantidad de definiciones de calidad [125-128], ya que es un concepto complejo con una gran cantidad de interpretaciones y por tanto difícil de definir. Se ha discutido en este sentido por décadas [129-134]. Detectando que el problema no es que el concepto de calidad sea subjetivo, sino que el problema es cómo poder relacionar los distintos puntos de vista sobre calidad en un mismo marco. Por ejemplo, un dominio de los conocimientos, una organización, un proyecto, etc. [49].

Concretamente, analizaremos las siguientes secciones. En la sección 2.4.1 se distingue la calidad de proceso y la calidad de producto. En la sección 2.4.2 se describe el estándar de calidad ISO/IEC 9126-1, sus antecedentes, funcionamiento y su utilización en la evaluación de la calidad del software, y en la definición y clasificación de requisitos. En la sección 2.4.3 se presentan las novedades aportadas por el estándar 25000:2005 respecto al 9126-1. Finalmente, en la sección 2.4.4 se presentan las aportaciones del grupo GESSI en calidad de software hasta el día de hoy.

### 2.4.1 Enfoques de Calidad de Software.

Los principales enfoques de como evaluar la calidad del software son [135]:

- La certificación del proceso de desarrollo de software que siguen las empresas que lo desarrollan. Algunas de las técnicas existentes son Capability Maturity Model (CMM) [136], Bootstrap [137], Capability Maturity Model Integration (CMMI) [138], Six sigma( $6\sigma$ ) [139] SPICE [140], ISO9000 [141].
- La evaluación del propio software mediante métricas que contrasten su calidad. En este enfoque realizar una especificación y una evaluación completa es un factor clave para garantizar la calidad adecuada.

El segundo enfoque nace principalmente, porque no existe la garantía de que un software obtenido mediante un proceso de desarrollo de calidad, sea un software de calidad. Esto es especialmente cierto en el marco de esta tesis de máster y por ello nos centramos en este enfoque, ya que para una empresa que desea seleccionar un componente OTS de calidad, lo realmente importante es que dicho software cumpla una serie de requisitos que dicha empresa ha definido. Estos requisitos los puede o no satisfacer un componente, independientemente del proceso seguido por sus desarrolladores.

La evaluación de la calidad de un producto de software puede lograrse mediante la definición apropiada de características de calidad, teniendo en cuenta el uso de dicho producto. Es importante que cada característica de calidad relevante del sistema de software sea evaluada, siempre que sea posible, utilizando medidas validadas y aceptadas, para ello hay que implementar un modelo de calidad de software.

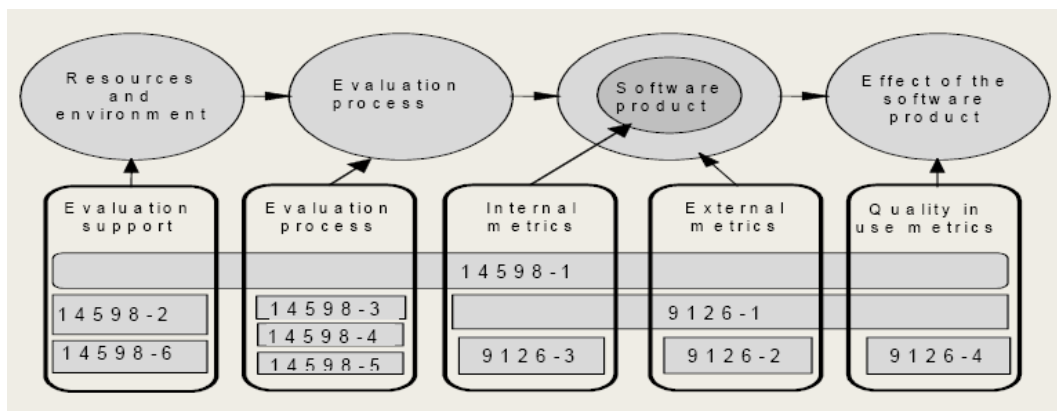
La organización internacional para la estandarización (ISO) [65] e IEC (the International Electrotechnical Commission) [142] ha definido una serie de estándares ISO y ISO/IEC relacionados con la calidad del software de producto.

ISO 9126 [143] es un estándar para la evaluación de la calidad de software, que junto con el estándar ISO/IEC 14598 que propone el uso de una serie de normas a seguir para el proceso de evaluación de un software, actualmente están siendo reemplazados por el ISO 25000:2005 [144], también llamado SQuaRE (Software product Quality Requirements and Evaluation), que sigue los mismos conceptos generales.

A continuación en la sección 2.4.2 se presenta el estándar ISO/IEC 9126-1, usado como se verá más adelante como soporte a esta tesis de máster. En el apartado ISO/IEC 25000:20052.4.3 se enumeraran las novedades aportadas por en nuevo estándar SQuaRE descartado en este trabajo por su aun con poca difusión y utilización.

## 2.4.2 ISO/IEC 9126-1

El estándar 9126 para calidad en productos software, es uno de los más extendidos y debe ser usado en conjunto con el ISO/IEC 14598 para la evaluación de productos software como se muestra en la figura 2-D.



**Figura 2-D: Utilización del estándar ISO 9126**

Otros estándares relacionados o que pueden ser usados en conjunto con el ISO/IEC 9126 y ISO/IEC 14598 son:

- ISO/IEC 12119: Requisitos de calidad para paquetes software
- ISO/IEC 12207: Proceso del ciclo de vida del software
- ISO/IEC 14143: Medidas de software
- ISO/IEC 15271: Guía para ISO/IEC 12207
- ISO/IEC 15504: Gravamen del proceso de software (conocido como SPICE)
- ISO/IEC 15939: Proceso de medición de software

El estándar ISO/IEC 9126:2001 consiste de cuatro partes: (ver figura 2-E)

- 9126-1: Modelos de Calidad [143]
- 9126-2: Métricas externas. las cuales miden el software en sí mismo (Calidad Externa)
- 9126-3: Métricas internas. las cuales miden el comportamiento del sistema (Calidad Interna)

- 9126-4: Calidad en el uso de métricas. el cual mide el efecto de usar el software en un contexto específico

El estándar ISO/IEC 9126:2001 contiene modelos de calidad y métricas. Debido a su naturaleza genérica, algunos de los conceptos presentados necesitan ser refinados antes de usar el estándar en un proyecto real.

Presenta tres diferentes vistas de calidad. [145] Las primeras dos son las vistas internas y externas que comparten las mismas 6 características y 26 sub-características, como vemos en la figura 2-F. La tercera vista es la calidad en uso, y tiene sus propias 4 características.

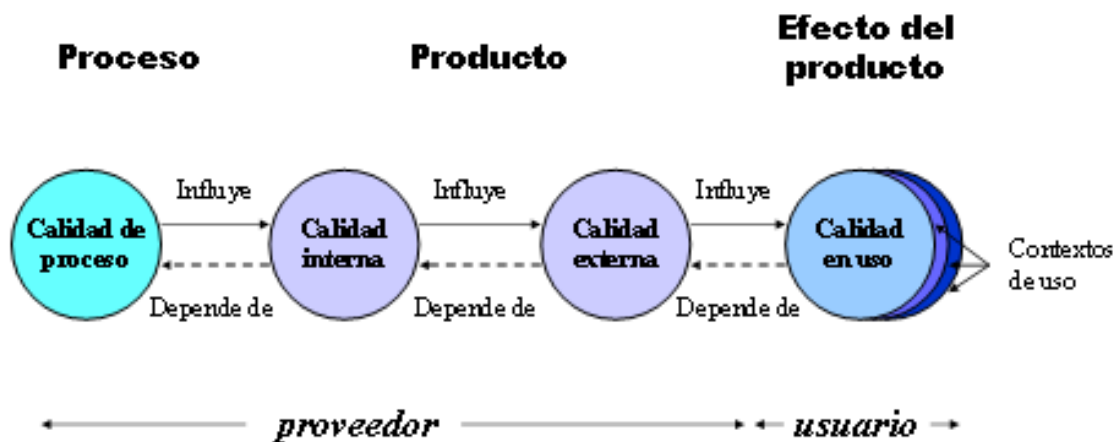


Figura 2-E: Partes del estándar ISO/IEC 9126

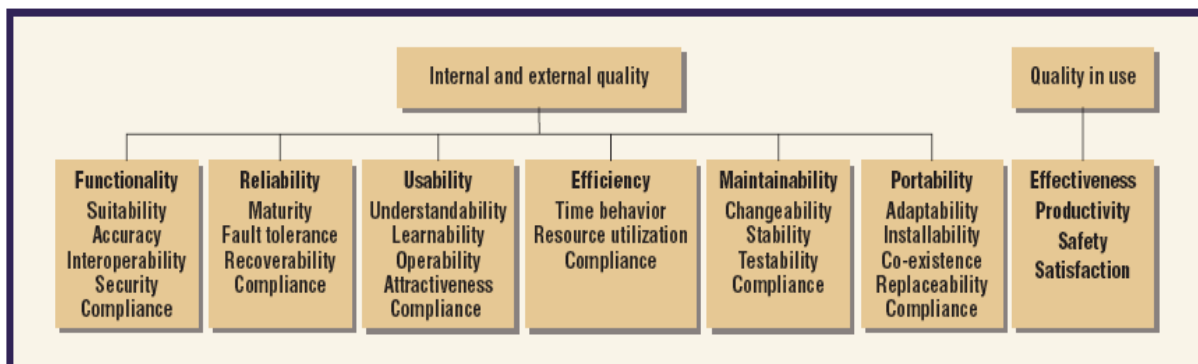


Figura 2-F: Vistas del estándar ISO/IEC 9126

La vista interna se refiere principalmente a propiedades estáticas de las partes individuales de los productos de software, incluido el diseño, la estructura y la complejidad de los elementos del código.

La vista externa se refiere al software terminado ejecutándose en el equipo de hardware con datos reales. En este punto de vista, los aspectos dinámicos del software juegan un papel importante.

La vista de calidad en uso se refiere al desempeño de usuarios específicos desempeñando tareas específicas con el software en su entorno real. Esta vista mide la productividad y la eficiencia de los usuarios finales.

Estas tres vistas se apoyan mutuamente, la vista interna influye en la externa que a su vez influye en la calidad de uso.

El ISO/IEC 9126-1 está definido por características generales de software, que son refinadas en sub-características; que a su vez se descomponen en atributos, pertenecientes a una jerarquía multinivel; pueden aparecer jerarquías de atributos y sub-características. Al final de la jerarquía aparecen los atributos de medición, quienes dan los valores para ser usados en la métrica.

Un atributo es una propiedad medible. Por ejemplo, tamaño, que puede ser medido como un número de líneas de código. Por lo que es posible determinar el comportamiento de un producto de software a través de sus propiedades inherentes y su calidad a través de sus atributos de calidad inherentes. Por lo que la medición de software se convierte en una liga entre el modelo de calidad y la calidad de software.

Esto implica que se puede especificar los requisitos de calidad del software proporcionando un conjunto de medidas de calidad relacionadas con los valores objetivo.

En la tabla 2-C se enumeran las seis características de calidad definidas en el modelo de calidad ISO/IEC 9621-1 y su descomposición en sub-características.

Characteristic/ Subcharacteristic		Description
1	<b>Functionality</b>	
	1	Suitability Presence and appropriateness of a set of functions for specified tasks
	2	Accuracy Provision of right or agreed results or effects.
	3	Interoperability Capability to the software products to interact with specified systems
	4	Security Prevention to (accidental or deliberated) unauthorized access to data
2	5	Functionality Compliance Adherence to application of functionality related standards or conventions
	<b>Reliability</b>	
	1	Maturity Capacity to avoid failures as a result of faults in the software
	2	Fault Tolerance Ability to maintain a specified level of performance in case of faults
	3	Recoverability Capability to re-establish level of performance after faults
3	4	Reliability Compliance Adherence to application of reliability related standards or conventions
	<b>Usability</b>	
	1	Understandability Effort for recognizing the logical concept and its applicability
	2	Learnability Effort for learning software application
	3	Operability Effort for operation and operation control
4	4	Attractiveness Capability of the product to be attractive to the user
	5	Usability Compliance Adherence to application of usability related standards and conventions
	<b>Efficiency</b>	
	1	Time behaviour Response and processing times; throughput rates
	2	Resource Utilization Amount of resources used and the duration of such use
5	3	Efficiency Compliance Adherence to application of efficiency related standards or conventions
	<b>Maintainability</b>	
	1	Analyzability Identification of deficiencies, failure causes, parts to be modified, etc.
	2	Changeability Capability to enable a specified modification to be implemented
	3	Stability Capability to avoid unexpected effects from modifications
6	4	Testability Capability to enable for validating the modified software
	5	Maintainability Compliance Adherence to application of maintainability related standards or conventions
	<b>Portability</b>	
	1	Adaptability Opportunity for adaptation to different environments
	2	Installability Effort needed to install the software in a specified environment
7	3	Coexistence Capability to co-exist with other independent software in a common environment sharing common resources
	4	Replaceability Opportunity and effort for using software in the place of other software
	5	Portability Compliance Adherence to application of portability related standards or conventions

Tabla 2-C: El modelo de calidad interno/externo ISO/IEC 9126-1

La valoración de estas características es útil para definir los requisitos del producto utilizando únicamente las características que se empleen en la práctica. Para algunos tipos de productos de software, hay determinadas características que no son significativas o no son garantía que con ellas se comprendan todos los requerimientos de los productos de software, por lo que en cada caso habrá que completarlas con otras definiciones más específicas para un producto o dominio específico.

### **2.4.3 ISO/IEC 25000:2005**

La independencia en los ciclos de vida del ISO/IEC 9621 y del ISO/IEC 14598 ha creado inconsistencias entre ellos, esto aunado a que tienen normativas, raíces, referencias y funcionales comunes, y a que forman un conjunto complementario de estándares, ha impulsado la creación de una nueva serie de estándares internacionales. Concretamente el nuevo estándar que nació con la idea de sustituir a estos dos es el estándar SQuaRE [144].

SQuaRE (es decir, la norma ISO/IEC 25000) contiene un modelo de calidad de software genérico que puede ser aplicado a cualquier producto de software, adaptado a un propósito específico.

El objetivo general para la creación de SQuaRE fue cubrir los procesos de especificación de requerimientos de calidad de software y evaluación de calidad de software, con el apoyo de procesos de medición de calidad.

El objetivo de SQuaRE es ayudar a aquellos que se encuentran en el proceso de desarrollo y selección de productos de software, estableciendo los criterios para la especificación de software, requisitos de calidad de los productos, su medición, y evaluación.

Incluye un modelo de calidad para alinear las definiciones de calidad de los clientes con los atributos de calidad del proceso de desarrollo. Además, proporciona medidas recomendadas para los atributos de calidad de los sistemas de software que pueden ser utilizados por los desarrolladores, adquirentes, y evaluadores.

Los beneficios encontrados con respecto a su antecesor incluyen:

- La coordinación de la guía en la medición y evaluación de la calidad de los productos de software
- Orientación para la especificación de requisitos de calidad de los productos de software, y
- Armonización con la norma ISO / IEC 15939, en forma de modelo de referencia en la medición de la calidad del producto presentado en la norma ISO / IEC 25020

Las principales diferencias con sus antecesores son:

- La introducción de un modelo de referencia general
- La introducción de guías detalladas y dedicadas a cada división
- La introducción de elementos de medidas de calidad dentro de la división de medición de la calidad
- Incorporación y revisión del proceso de evaluación
- Coordinación y Armonización de el contenido de ISO/IEC 15939

SQuaRE consiste de 11 documentos agrupados en 5 divisiones (familias de estándares) presentados en la tabla 2-D. Estos documentos conforman la arquitectura de SQuaRE (ver figura 2-G).

<b>Requisitos de Calidad 2503n</b>	<b>Modelo de Calidad 2501n</b>	<b>Evaluación de Calidad 2504n</b>
	<b>Gestión de Calidad 2500n</b>	
	<b>Medición de Calidad 2502n</b>	

**Figura 2-G: Arquitectura de SQuaRE**

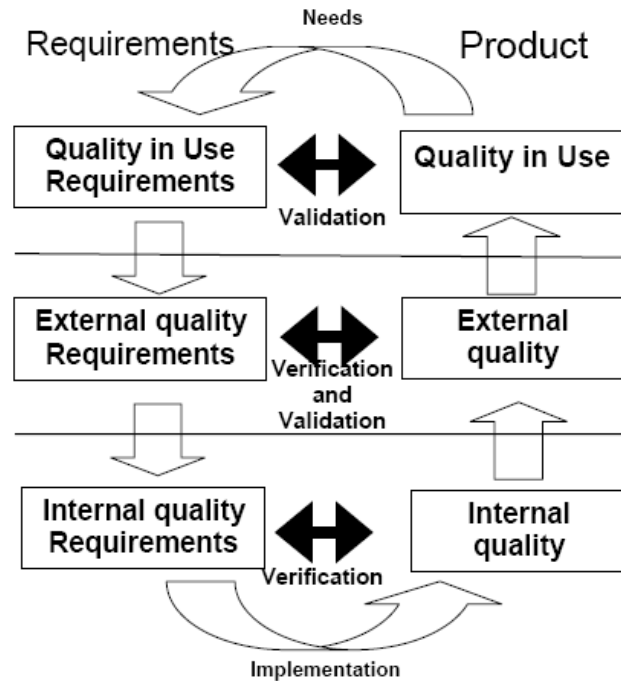
<b>Estándar</b>	<b>División</b>	<b>Descripción</b>
ISO/IEC 2500n	Gestión de calidad	Este estándar define todos los modelos, términos y definiciones comunes en la serie SQuaRE. Guiando a los usuarios a través de los documentos SQuaRE, incluye además sugerencias para ayudar a los usuarios en la aplicación del estándar adecuado a aplicaciones específicas. Provee además requisitos y guías como apoyo al responsable de la gestión de las especificaciones de los requisitos y la evaluación del producto de software
ISO/IEC 2501n	Modelo de calidad	Esta división incluye un modelo de calidad detallado basado en ISO/IEC 9126, comprendiendo características de calidad internas y externas y para calidad en uso. Además, el modelo descompone estas características en sub-características. Esta división también incluye un modelo de calidad de datos.
ISO/IEC 2502n	Medición de calidad	Esta división incluye un modelo de referencia para medir la calidad de un producto de software, definiciones matemáticas de las medidas de calidad y la guía práctica para su aplicación, estas medidas se aplican tanto a calidad interna y externa como calidad en uso.
ISO/IEC 2503n	Calidad de requisitos	ISO/IEC 25030 es el único estándar en esta división, ayuda a identificar y especificar requisitos de calidad; los desarrolladores pueden usar estos requisitos de calidad para elicitar y definir requisitos de calidad para el producto de software que va a desarrollarse o como entrada para un proceso de evaluación.
ISO/IEC 2504n	Evaluación de calidad	Estas normas contienen los requisitos, recomendaciones y guías para la evaluación de productos de software, ya sea realizado por evaluadores, adquirentes, o desarrolladores independientes (internamente en el desarrollo de la organización). También presenta ayuda para documentar la medida como una evaluación del módulo. Esta división se basa en la serie de estándares ISO / IEC 14598.

**Tabla 2-D: Familia de estándares SQuaRE**

Además posee una extensión (ISO/IEC 25050 a la ISO/IEC 25099) diseñada para contener estándares internacionales de calidad en productos de software y reportes técnicos que se ocupan de dominios de aplicación específicos o que se pueden usar como complemento de uno o mas estándares internacionales SQuaRE.

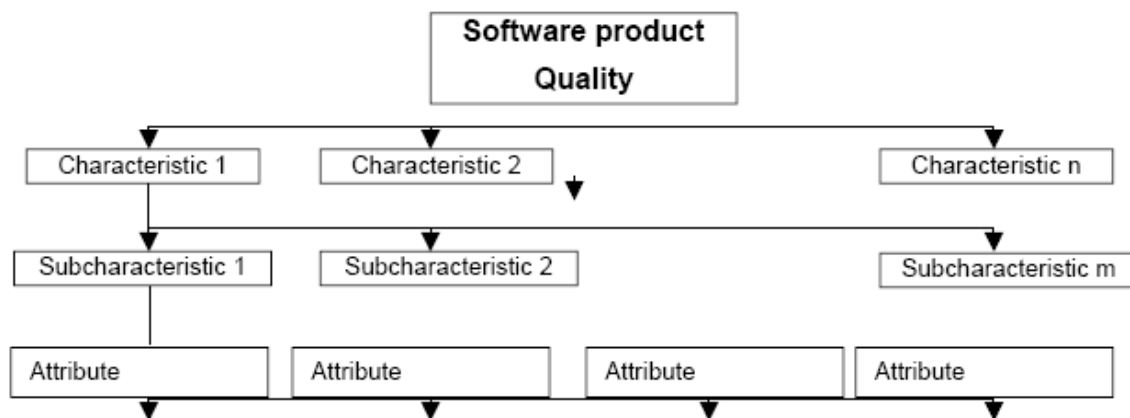
En la figura 2-H se presenta un modelo del ciclo de vida de la calidad del producto software y se identifican tres fases principales:

- Producto en desarrollo,
- Producto en operación y
- Producto en uso.



**Figura 2-H: Ciclo de vida de la calidad de un producto de software**

En la figura 2-I se presenta la estructura del modelo de calidad categoriza la calidad de software en características, que a su vez se dividen en sub-características y atributos de calidad.



**Figura 2-I: Estructura del modelo de calidad**



CURRENT		SQuaRE
9126: Product quality		25000: Quality Management Division
-1: Quality model		25000: Guide to SQuaRE (NP)
-2: External metrics		25001: Planning and management
-3: Internal metrics		25010: Quality Model Division
-4: Quality in use metrics		25010: Quality model (Rev)
		25020: Quality Measurement Division
New Proposal		25020: Measurement reference model and guide (NP)
Guides to use 9126 & 14598		25021: Quality measure elements
Base metrics		25022: Measurement of internal quality
Quality requirements		25023: Measurement of external quality
		25024: Measurement of quality in use
14598: Product evaluation		25030: Quality Requirements Division
-1: General overview		25030: Quality requirements (NP)
-2: Planning and management		25040: Quality Evaluation Division
-3: Proc for developers		25040: Quality evaluation reference model and guide
-4: Proc for acquirers		25041: Evaluation modules
-5: Proc for evaluators		25042: Process for developers
-6: Doc of evaluation modules		25043: Process for acquirers
		25044: Process for evaluators

Figura 2-J: Relación de SQuaRE con ISO/IEC 9126 e ISO/IEC 14598

En la figura 2-J se muestra la relación de SQuaRE con ISO/IEC 9126 e ISO/IEC 14598.

En donde:

- La columna titulada Actual, enlista todos los estándares existentes de las series ISO/IEC 9126 e ISO/IEC 14598 y los nuevos estándares propuestos.
- La columna SQuaRE enlista los estándares pertenecientes a esta serie.
- Las flechas precisan las relaciones entre los estándares, indican el proceso de transición ya que algunos de los nuevos estándares son resultado de la concatenación, unificación y revisión de más de un documento de las series anteriores.

#### 2.4.4 Calidad de software en el grupo GESSI

El ISO/IEC 9126-1 se toma como base para muchos de los métodos y herramientas aportados por GESSI.

En [146] se muestra como y cuales conceptos deben perfeccionarse antes de utilizar la norma en un verdadero proyecto y se destaca la necesidad de contar con herramientas de apoyo para la creación de modelos de calidad.

En [147] se aborda la construcción de modelos de calidad de software de dominio complejo, que se define como ámbitos que implican una combinación de funcionalidades. [148] Propone una metodología para definir modelos de calidad basada en el estándar ISO / IEC 9126-1 para el dominio de servidores de correo. [149] Propone un catálogo de características no técnicas, diseñado para integrarse sin problemas o para ampliar la que se

ha incluido en el estándar ISO / IEC 9126-1 una serie de posibles aplicaciones del catálogo resultante, destinada a apoyar diversas actividades del ciclo de vida de sistemas basados en componentes COTS. COSTUME [150] es un método para construir modelos de calidad para sistemas de software basados en componentes. Estos sistemas están compuestos de varios componentes COTS interconectados, por lo que se necesita seleccionar más de un componente y por tanto de un modelo de calidad más complejo. [151] construye un modelo de calidad basado en ISO/IEC 9126-1 mediante el uso de jerarquías de categorías de dominios de componentes COTS. En [152] se presenta un lenguaje para expresar la calidad de un componente en el marco del estándar ISO/IEC.

Con el fin de formalizar las cuestiones de calidad de software [153] propone un modelo genérico que representa los conceptos fundamentales relacionados con la calidad del software [154] integra esta propuesta en el meta-modelo de UML y, en concreto, en la arquitectura MOF de cuatro niveles

En [155] se revisan los problemas sobre la utilización de modelos de calidad en la práctica y se propone un conjunto de características que brindan la base para la construcción de modelos de calidad útiles y bien fundados. Basándose principalmente en componentes COTS

QM es una herramienta que sirve para apoyar al método IQMC [156, 157] para la construcción de modelos de calidad en dominios componentes COTS. Esta herramienta también apoya la organización de modelos de calidad en una taxonomía de dominios de componentes COTS [158]

En [159] se propone la adopción de modelos de calidad como medio para estructurar la descripción de las capacidades en un componente COTS particular enterprise resource planning (ERP). Así como el uso de estos modelos como forma de procesamiento de los requisitos de calidad en los ERP.

## 2.5 Ingeniería de Dominio

En esta parte del estado del arte se describen las principales características de la ingeniería de dominio y más concretamente de una de las actividades que abarca, que es el análisis del dominio. Vemos también la relación entre el análisis de dominio y la elicitación de requisitos de un sistema informático. Finalmente, presentamos una de las técnicas existentes de análisis de dominio y un lenguaje apropiado para representarlo.

Los procesos de desarrollo basados en la reutilización de software se clasifican en [160]:

- **Desarrollo para reutilización.** Este tipo de desarrollo, se hace pensando en la reutilización, es decir, en la adaptación o construcción de componentes con el propósito de ser reutilizados en futuras aplicaciones.
- **Desarrollo con reutilización.** Este tipo de desarrollo consiste en la generación de nuevos productos software integrando y reutilizando un conjunto de componentes existente de forma directa o pasando por un proceso de adaptación.

Apareciendo así dos tipos de ingeniería (ver figura 2-K):

**Ingeniería de dominio:** La ingeniería de dominio [161] se centra en análisis, especificación e implementación de activos reutilizables de software relativos a un dominio. Estos activos pueden ser modelos, arquitecturas, componentes y artefactos de software en general, que podrán ser usados en el desarrollo de múltiples productos de software relacionados con dicho dominio. Para facilitar la identificación de los activos a reutilizar, la ingeniería de dominios se pueden usar repositorios que permiten la organización, selección y mantenimiento de dichos activos [162].

**Ingeniería de aplicación:** la ingeniería de aplicación [161] se orienta hacia la construcción o desarrollo de productos individuales que satisfacen un conjunto de requisitos y restricciones (expresados por un usuario específico), basándose en la reutilización de componentes existentes y en el conocimiento del dominio.



**Figura 2-K: Reutilización de Software**

A continuación nos centramos únicamente en la ingeniería de dominio ya que es la que esta relacionada con la tesis de máster que aquí se presenta.

La ingeniería de dominio abarca tres actividades principales [161-163] (ver figura 2-L):

- Análisis de dominio. Proceso de generación de conocimiento.
- Diseño del dominio. Especificación de la infraestructura.
- Implementación del dominio. La implementación de la infraestructura.

Concretamente, en esta tesis de máster nos interesa únicamente la actividad de análisis de dominio. En la sección 2.6.1 se describe el estado del arte de la investigación relacionada con dicha actividad. En la sección 2.6.2 se presentan los modelos orientados a actores y objetivos, que son dos posibles tipos de modelos resultantes del análisis de dominio. Finalmente, en la sección 2.5.3 se presenta el lenguaje *i\** que es uno de los existentes para representar modelos orientados a actores y objetivos.

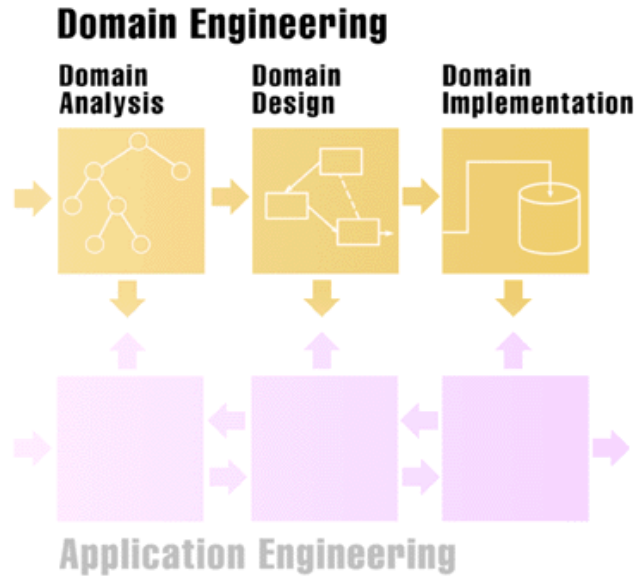


Figura 2-L: Actividades de la Ingeniería de Dominio

### 2.5.1 Análisis del Dominio

El análisis de dominio fue definido por primera vez por [164], posteriormente [165] lo definieron como el proceso de identificar, recopilar, organizar y representar la información relevante de un dominio con el propósito de hacerla reutilizable.

Una de las diferencias respecto al análisis tradicional, es que el análisis de dominio se hace considerando un conjunto de sistemas o aplicaciones que pudieran resultar ser componentes de una futura aplicación.

Durante este proceso se estudia distintas fuentes de información relativa al dominio, en donde se identifican características comunes en las aplicaciones de dicho dominio.

Estas fuentes de información pueden ser:

- Revistas especializadas, manuales del producto, tutoriales, demos, reportes técnicos, casos de estudio publicados, white-papers, portales de Internet como eCots [166], INCOSE[167], etc.
- Informes técnicos y estándares en particular proporcionados por grupos de los principales proveedores de componentes OTS de un dominio particular. Por ejemplo, el Internet Mail Consortium (IMC) [168] en el caso de sistemas de e-mail, el Workflow Management Coalition (WfMC) [169] en el caso de tecnologías de flujo de trabajo, o Enterprise Content Management Association (AIIM) [170] en el caso de tecnologías de administración de documentos.
- Implementaciones y aplicaciones ya existentes así como toda la información disponible para estas, considerando datos actuales y futuros en caso de evoluciones del producto [171]

Durante el estudio, la experiencia y conocimiento son acumulados hasta alcanzar el punto cuando la abstracción puede ser sintetizada y puesta a disposición para su reutilización. Se

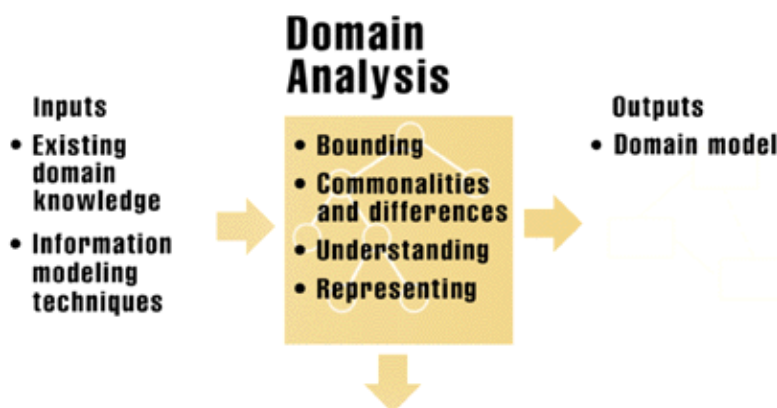
trata de un proceso de aprendizaje lento y no-estructurado que lleva a la identificación, abstracción, y encapsulamiento de objetos del dominio de estudio.

Concretamente, las tareas que se realizan son las siguientes [172]:

- Limitación del dominio a considerar
- Identificación de las similitudes y diferencias entre los elementos del dominio.
- Razonamiento y comprensión de las relaciones entre los distintos elementos en el dominio.
- Representación este conocimiento de una forma útil.

La información obtenida a partir de este estudio se representará como un modelo del dominio. Los ingenieros de software o de aplicación utilizarán uno de estos modelos cuando necesiten seleccionar un componente a reutilizar en un dominio. En ellos podrán encontrar información que les facilitará la comprensión del dominio.

La figura 2-M representa estas tareas, así como las entradas necesarias para su realización y las salidas obtenidas como resultado [161].



**Figura 2-M: Tareas relativas al Análisis de Dominio**

El objetivo del análisis de dominio por tanto, es producir un modelo de dominio que puede servir como [171]:

- Fuente de referencia unificada para cuando surjan ambigüedades durante el análisis de problemas, o después durante la implementación de componentes reutilizables.
- Repositorio compartido de los conocimientos para la enseñanza y la comunicación.
- Especificación para el desarrollador de componentes reutilizables.

### **2.5.2 Modelado Orientado a Objetivos y el Modelado Orientado a Actores**

En los últimos años, la construcción de modelos orientados a objetivos y actores se ha convertido en una herramienta habitual en distintos ámbitos tales como el de la ingeniería de requisitos, y el del modelado de procesos en organizaciones [173].

Estos modelos tienen particular relevancia desde el punto de vista de la ingeniería de requisitos y en el análisis de dominio. Es de destacar la facilidad con la que permiten describir un dominio y su contexto, con los actores que en él intervienen, los objetivos de

cada actor y las dependencias entre estos actores. A partir de estos modelos es posible determinar parte de los requisitos de un sistema software correspondiente a dicho dominio.

Estos dos tipos de modelos están muy cercanos entre si:

- Los modelos orientados a objetivos proporcionan una descomposición y refinamiento de las necesidades del usuario en objetivos concretos que resulta muy útil en las fases preliminares del análisis de requisitos.
- Los modelos orientados a actores extienden el modelado orientado a objetivos, permitiendo modelar el sistema mediante una red de actores y dependencias.

La principal diferencia entre el modelo orientado a objetivos y el modelo orientado a actores radica en:

- El modelado orientado a objetivos tiene un actor central para el cual se delimita el dominio, abarcando únicamente lo correspondiente al dominio de estudio; identificando aquellos actores con los cuales tiene algún tipo de dependencia.
- Por su parte en el modelo orientado a actores existen múltiples actores pertenecientes al mismo nivel y dominio de estudio, con los cuales se establecen las relaciones y dependencias. Creando con ello una red de dependencias y relaciones entre actores.

A continuación describiremos el uso de los dos tipos de modelos en el ámbito de la ingeniería de requisitos.

### 2.5.2.1 Modelado Orientado a Objetivos

Estos modelos se utilizan en las actividades básicas de la ingeniería de requisitos concretamente en las actividades de elicitación, elaboración, verificación, validación, explicación y negociación [173].

Un objetivo es una meta a alcanzar por el sistema, por lo tanto, la formulación de objetivos ayudará a determinar los requisitos del sistema destinados a garantizar este logro.

Existen varias razones por las cuales es importante el uso de objetivos en ingeniería de requisitos [173].

- **Obtención de requisitos completos:** Los objetivos proporcionan los criterios precisos para complementar satisfactoriamente la especificación de requisitos; la especificación es completa con respecto a un conjunto de objetivos si puede probarse que todos los objetivos pueden ser alcanzados desde la especificación y desde las propiedades conocidas sobre el dominio.
- **Eliminación de requisitos irrelevantes:** Los objetivos proporcionan criterios precisos para los requisitos pertinentes; un requisito es pertinente con respecto a un conjunto de objetivos en el dominio considerado, si su especificación es usada en la consecución de por lo menos un objetivo.
- **Explicación de requisitos a los stakeholders:** Los objetivos proveen información racional de los requisitos. Los requisitos aparecen debido a que algunos objetivos subyacentes proveen una base para esto [14].

- **Estructuración de Requisitos:** El refinamiento de objetivos provee un mecanismo natural no solo para estructurar y mejorar en la legibilidad de libros de requisitos complejos, sino también para la exploración de varias alternativas.
- **Detección de conflictos en requisitos:** Los objetivos han sido reconocidos por ser la base para la detección de conflictos entre requisitos y resolverlos eventualmente.
- **Evolución de requisitos y trazabilidad:** La separación de información estable de la información volátil es otra preocupación importante en la gestión de la evolución. Un requisito representa una manera particular de alcanzar un objetivo en concreto; el requisito tiene por lo tanto más posibilidades de evolucionar hacia otra forma de alcanzar ese mismo objetivo que el objetivo en si. Por lo tanto podemos decir que los objetivos son más estables con respecto a los cambios [174].
- **Identificando requisitos.** Los objetivos conducen la identificación de requisitos para apoyarlas [175].
- **Conduciendo el refinamiento y la abstracción.** Una vez que se obtiene y valida un conjunto preliminar de objetivos y requisitos, muchos otros objetivos se pueden identificar mediante el refinamiento y la abstracción simplemente preguntando como y por que a los requisitos y objetivos disponibles.

Por todas ellas, podemos decir que un objetivo describe una meta que debe alcanzar un sistema [176]. Los objetivos pueden clasificarse en objetivos funcionales y de calidad. Los objetivos pueden ser refinados jerárquicamente en sub-objetivos a través de diversas estrategias, tales como el refinamiento y la descomposición, las decisiones de diseño y el análisis de resultados de un dominio específico. El proceso de refinamiento de objetivos finaliza cuando se asegura que todos los objetivos señalados serán cumplidos mediante los requisitos definidos.

Algunas técnicas de modelado orientado a objetivos y los lenguajes asociados [177] son las siguientes:

- **KAoS** (Knowledgeable Agent-oriented System) [178]. Permite construir modelos de requisitos a partir de los objetivos organizacionales. Esta aproximación está soportada por un marco formal que define cada término de forma rigurosa. La principal contribución de KAoS es la demostración de que los requisitos se corresponden con las metas del futuro sistema.
- **NFR** (Non-Functional Requirements) [179]. Es un marco cualitativo enfocado en la representación y razonamiento de requisitos no funcionales.
- **GRL** (Goal-oriented Requirement Language) [180]. Es un lenguaje para apoyar el modelado orientado a objetivos y de razonamientos con requisitos no funcionales. Forma parte de la notación URN (User Requirements Notation) [181], que ha sido propuesta como estándar ITU-T (International Telecommunication Union – Telecommunication Standardization Sector).[182] GRL distingue tres principales categorías de conceptos: elementos intencionales, relaciones intencionales y actores (que no admiten especializaciones). GRL ofrece constructores para establecer relaciones con elementos externos al modelo (elementos no-intencionales y atributos de conexión), y posee elementos adicionales de justificación y/o contextualización como creencias (*beliefs*), correlaciones, tipos de contribuciones y etiquetas de evaluación para especificar los estados de satisfacción.

### 2.5.2.2 Modelado Orientado a Actores

Los enfoques orientados a actores se han vuelto populares en ingeniería de software, tanto para el marco arquitectónico como para marcos de modelado de ingeniería de requisitos y diseño.

El modelado orientado a actores es altamente efectivo al representar requisitos y dar respuesta a cuestiones como los principales objetivos de los actores claves que intervienen en el sistema, y como cada uno de estos actores dependen de los demás.

Un modelo de dependencias entre actores contiene dos tipos de elementos: los actores y las dependencias entre ellos.

Los *actores* son entidades intencionales de manera que existe un motivo racional detrás de cada actividad que realizan. Los actores pueden ser especializados en: roles (roles) y agentes (agentes). Según [183], un *rol* es una caracterización del comportamiento de un actor social en un determinado contexto o dominio, mientras que un *agente* es un actor con manifestaciones concretas y físicas que juega un determinado rol.

Las *dependencias* conectan los actores fuente y destino, llamados *dependen* y *dependee* respectivamente. De esta manera se forma una red de conocimiento que permite entender "por qué" el sistema se comporta de una forma determinada [184].

Algunas técnicas de modelado orientado a actores y los lenguajes asociados descritos en [177]:

- *i\** [183]. Se ha propuesto para la elaboración de modelos de procesos de negocio y reingeniería. Los procesos son vistos como los sistemas sociales compuestos por actores que cooperan para alcanzar objetivos. El marco ofrece dos tipos de modelos de dependencia: un modelo de dependencia estratégica, utilizado para la descripción de procesos como redes de dependencias estratégicas entre los actores, y un modelo de la lógica estratégica utilizado para describir el razonamiento de cada uno de los actores en el proceso, así como para el estudio de otras estructuras proceso
- Tropos [185]. Es un proyecto en el que participan distintas universidades que tiene como objetivo definir una metodología de desarrollo de sistemas software orientados a agentes y que utiliza como lenguaje de modelado una variante de *i\** [186]. La metodología propuesta abarca desde la etapa de análisis de requisitos hasta la de implementación. En cada una de ellas se utilizan los conceptos propios de *i\** (actor, dependencia, etc.) para mostrar una vista del modelo teniendo en cuenta la etapa. Modela explícitamente de forma separada aspectos relacionados con el dominio de aspectos relacionados con el sistema software.
- ALBERT [187]. cuyo nombre significa lenguaje orientado a actores para construir y elicitar requisitos en tiempo real, es un lenguaje forma para la especificación de requisitos centrado en actores

Los objetivos, representan los intereses estratégicos de los actores.



Algunas aplicaciones de estos modelos son:

*i\** es utilizado en [14] para representar el conocimiento en notaciones gráficas. Separa la fase inicial de la fase tardía de los requisitos e ilustra el tipo de características de modelado y las capacidades de razonamiento que podrían ser apropiadas para la fase inicial de ingeniería de requisitos. Ayuda a entender el contexto organizacional y las justificaciones (los por qué).

El objetivo global de la investigación en [188] es realizar una evaluación del potencial de los lenguajes de modelado intencional, como BMM [189] e *i\**, en el contexto de arquitecturas de empresa, para evaluar los prospectos y los desafíos de incorporar conceptos de modelado intencional en la práctica de EA.

En [190] se exploran servicios electrónicos comerciales desde una perspectiva de objetivos estratégicos y de rentabilidad, mediante el trabajo conjunto de dos técnicas de ingeniería de requisitos, *i\** y e3value. *i\** ayuda en representar y analizar modelos de negocios de servicios, explorando objetivos empresariales estratégicos. e3value ayuda a ver como estos objetivos pueden resultar rentables.

En [191] el uso de *i\** apoya la gestión de la propiedad intelectual. En *i\**, los actores tienen objetivos así como el know-how y los recursos para alcanzar esos objetivos. Las Patentes restringen el uso de know-how, permitiendo a los actores posicionarse en una red de relaciones de dependencia. Para ilustrarlo muestra ejemplos del dominio de e-comercio.

Tropos es usado en [192] para construir un sistema que facilite el acceso de las industrias en Trento a los servicios logísticos disponibles en la web. La información obtenida es usada para motivar y definir los requisitos funcionales del sistema que se va a implementar en la organización.

En [193] se presenta una extensión de tropos mediante un marco formal para modelar y analizar requisitos de seguridad y confianza. Se analiza una organización construyendo un modelo de confianza determinando las relaciones de confianza entre los actores y creando un modelo funcional donde se compara el modelo de confianza contra las delegaciones reales.

En [194] se presenta un marco para la modelación de las relaciones entre los actores estratégicos con el fin de obtener, identificar y analizar los requisitos de seguridad. En particular, el análisis de dependencias entre actores ayuda en la identificación de atacantes y sus posibles amenazas. El análisis de objetivos ayuda a elicitar el proceso dinámico de toma de decisiones del sistema de jugadores para las cuestiones de seguridad. Se presenta para ilustrar el marco propuesto un ejemplo de P2P.

En [195] se describe un enfoque basado en técnicas de modelación intencional. El análisis intencional se presenta en *i\**. Este enfoque ayuda a desarrollar tecnologías para apoyar la gestión del conocimiento distribuido. Se presentan dos ejemplos de caso estudio de un hospital para ilustrar.

Producir requisitos de usuario final para el rediseño del sistema de control del proceso de información de vuelos usado en el aeropuerto de Atenas. Presenta la metodología empleada para la elicitación.

### 2.5.2.3 El lenguaje $i^*$

El lenguaje  $i^*$  [183, 196] es la propuesta seminal de la familia de modelos orientados a objetivos y actores.  $i^*$  es una de las notaciones más empleadas para construir este tipo de modelos.

El éxito de estos trabajos fue tal que a día de hoy existe un sitio web oficial [197] y un sitio colaborativo wiki [198].

El marco  $i^*$  fue propuesto por Eric Yu, Mylopoulos y Chung. en los años 90 [183], y fue desarrollado para modelar y hacer razonamientos sobre entorno organizacionales y sus sistemas de información.

$i^*$  permite expresar de forma clara y sencilla los objetivos de los actores que aparecen en los modelos y las dependencias entre ellos. Además,  $i^*$  cuenta con una notación gráfica que permite tener una visión intuitiva y unificada del entorno modelado mostrando tales actores y dependencias

$i^*$  se propuso originariamente para modelar y razonar sobre sistemas organizacionales y sistemas de información. Su notación se puede aplicar a distintas entidades (organizaciones, sistemas informáticos, etcétera), es sencillo y clarifica enormemente la noción de quien necesita qué. Durante la construcción de un modelo basado en  $i^*$  se deben detectar los actores y sus necesidades u objetivos. Estas necesidades deben ser cubiertas por otras partes del sistema o por otros sistemas. El buen funcionamiento del sistema informático (en este caso) se basa en que todas las necesidades queden cubiertas, para ello en el  $i^*$  permite indicar tipos distintos de necesidades.

Los modelos que se pueden representar en  $i^*$  son de dos tipos *modelos estratégicos de dependencias* (SD) y *modelos estratégicos de razonamiento* (SR).

Un modelo SD describe una configuración particular de las relaciones y dependencias entre actores, entendiendo por actores las unidades o partes de una organización. Consisten en un conjunto de nodos que representan actores (*actors*) y un conjunto de relaciones entre estos actores. Las relaciones representan dependencias (*dependencies*) que permiten expresar que un actor (depend) depende de otro actor (*dependee*) para conseguir un determinado propósito (*dependum*). El *dependum* es un elemento intencional que puede ser de tipo: recurso (*resource*), tarea a realizar (*task*), objetivo (*goal*) o requisito no funcional (*softgoal*). El modelo permite definir la importancia (*strength*) de la dependencia para cada uno de los actores que intervienen en ella.

Un modelo SR describe la descomposición de los objetivos de los actores. Permitiendo visualizar los elementos intencionales dentro de los límites (*boundary*) de un actor para refinar el modelo SD y añadirle capacidad de razonamiento. Para ello, se asignan las relaciones de dependencia del SD a elementos intencionales internos al actor y se añaden relaciones medio-fin (*means-end*) y descomposición de tarea (*task-decomposition*): una relación *means-end* se establece entre dos elementos intencionales de un mismo actor (uno de ellos es el medio, normalmente una tarea, que contribuye a la realización del otro que es un fin); una relación *task-decomposition* descompone una tarea en elementos intencionales de cualquier tipo.

Los actores pueden especializarse en agentes (*agents*), roles (*roles*) y cargos (*positions*). Un cargo puede cubrir varios roles. Los agentes representan instancias particulares de personas, máquinas o software que ocupan un determinado cargo dentro de la organización y que, por lo tanto, pueden jugar varios roles. Los actores y sus especializaciones pueden desagregarse en otros actores mediante la relación *Is-Part-Of*.

La figura 2-N nos sirve para ilustrar con un ejemplo de sistema de tutoría académica el uso de  $i^*$  y los conceptos previamente expuestos. En donde, a la izquierda se muestra el SR del tutor y las relaciones que se establecen entre sus elementos intencionales internos. A la derecha se muestra el SD del alumno con sus dependencias respecto al tutor [199].

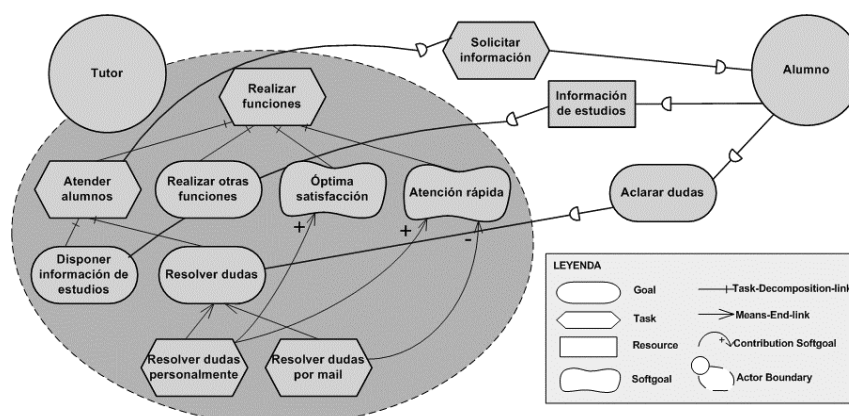


Figura 2-N: Modelo  $i^*$  de Yu para un proceso de tutoría académica

Desafortunadamente, no existe una definición única de  $i^*$  sino diferentes versiones y variantes que han sido comparadas en diversos estudios [177, 196, 199-202].

### 2.5.3 Aportación de GESSI a $i^*$

En el grupo GESSI, se realiza el análisis de dominio utilizando principalmente  $i^*$ . Dicho análisis permite estructurar la información y conocimiento obtenidos sobre un dominio a fin de que pueda ser reutilizada eficientemente. En [203] se enfrenta dos problemas básicos, los equipos multidisciplinarios y la alineación de los objetivos que los diferentes participantes puede tener durante el proceso. Tratando el alineamiento mediante su definición en  $i^*$ .

Debido a la flexibilidad de  $i^*$ , puede llegar a ser difícil su utilización. Por lo que en [199, 204] se presentan varias técnicas de modelado para  $i^*$  comparándolas en base a un conjunto de criterios. Mientras que [205] presenta un modelo conceptual genérico para ser utilizado como marco de referencia de las variantes de  $i^*$ .

En [206] se propone un marco para la definición de métricas sobre modelos de dependencias entre actores con el objetivo de facilitar el análisis de ciertas propiedades de los procesos tales como la seguridad, la eficiencia o la precisión. Se utilizan modelos de dependencia  $i^*$  y diagramas de actividad de UML como estrategia para modelar el proceso de implantación de Balanced Scorecard (BSC) [207]. En [208] describe la aplicación experimental de  $i^*$  para modelar una arquitectura de sistema en términos de dependencias

entre componentes. [209] Se extiende Tropos para su implementación en PROLOG añadiendo un paso que permite incorporar decisiones relevantes en tiempo de diseño.

También se ha analizado la semántica de  $i^*$  realizando mejoras al respecto. En [210] se da una definición precisa del concepto de herencia para la elaboración de modelos en  $i^*$  con un enfoque en el modelo (SR), mientras los elementos de extensión, refinamiento y redefinición son definidos en [211]. [212] Propone un enfoque que complementa el marco  $i^*$  con modelos de decisión mediante modelado de variabilidad ortogonal. RiSD [213] incrementa la comprensión de los métodos orientados a metas mientras mejora su construcción. Es una metodología para la construcción de modelos de dependencia estratégica (SD) en  $i^*$ .

Se propone el uso de herramientas tal como J-PRiM [214] que consiste en una herramienta java permitiendo definir modelos  $i^*$  mediante la aplicación de PRiM [215] (Metodología de Reingeniería de Procesos  $i^*$ ). REDEPEND-REACT [216] para apoyar la definición y evaluación de propiedades arquitectónicas extendiendo el plug-in de Microsoft Visio REDEPEND. Este plug-in modela gráficamente objetivos mediante  $i^*$ .

## 2.6 ERP

Un Enterprise Resource Planning (ERP) es un componente OTS que proporciona una plataforma de tecnología mediante la cual las organizaciones pueden integrar y coordinar sus principales procesos internos de negocios. Cada organización tiene necesidades distintas y los ERP necesitan una parametrización que depende de estas necesidades [217].

La implantación de sistemas de gestión ERP en las empresas no es un fenómeno nuevo. Por lo que a lo largo de los años ha pasado por diferentes etapas basadas en la idea de incorporar mayor funcionalidad.

Existe una gran cantidad de casos de éxito reportados y se aportan muchas ventajas gracias a su implementación. Sin embargo las investigaciones demuestran que mientras unas organizaciones tienen éxito en su implementación, otras tienen que aceptar beneficios mínimos o abandonarla [218], e incluso algunas caen en bancarrota por perder el control en la implementación [219].

Un problema comúnmente encontrado es el alineamiento en la adaptación del ERP [217, 219], que consiste en alinear los requisitos de los stakeholders con los del ERP. Esta necesidad de alineamiento nos lleva al problema común de los componentes OTS: La selección apropiada y correcta del componente. En este caso un ERP.

En este apartado exploraremos el estado del arte en la selección de ERP dividiendo el capítulo en 5 secciones: En la sección 2.6.1 se habla de la historia de los ERP, su evolución y las perspectivas futuras. En la sección 2.6.2 se presenta la definición de los ERP y sus características. En la sección 2.6.3 se presentan los módulos que conforman a los ERP y sus funcionalidades. En la sección 2.6.4 se presenta la relación de los ERP con otros sistemas empresariales. En la sección 2.6.5 se presenta el ciclo de vida de los ERP, los casos de éxito y fracaso, la importancia en la selección de ERP y los métodos existentes. Y por último, en la sección 2.6.6 se describe las principales aportaciones realizadas por el grupo GESSI.

### 2.6.1 Historia

El surgimiento de los ERP [220, 221] data de 1960 con la introducción en entornos industriales del Inventory Management & Control. En ese momento el principal uso de software en dichos entornos era para la gestión de inventario, y la mayor parte del software utilizado era hecho a medida y diseñado según los conceptos tradicionales de gestión de inventarios.

En 1970s surgieron los llamados Material Requirement Planning (MRP). Con ellos se pretendía planificar los materiales que serían necesarios durante el proceso de producción, y gestionar la adquisición de dichos materiales.

En 1980s evoluciona hacia los sistemas Manufacturing Requirements Planning (MRP II). Estos sistemas incluían la gestión de la planta de fabricación y actividades relacionadas con la distribución de los artículos fabricados, sincronizando los materiales con requisitos de producción.

En 1990s los MRP-II fueron ampliados, aun más, para abarcar áreas como Ingeniería, Finanzas, Recursos Humanos, Gestión de Proyectos, etc.; es decir la totalidad de las funciones desarrolladas dentro de una empresa. Fue esta evolución la que introdujo el concepto de ERP.

El mercado de los ERP creció rápidamente en los 90s entre otras razones debido a que:

- El enfoque cliente/servidor se hizo popular en las empresas y los ERP fueron diseñados para tomar ventaja de ello [219].
- Las implementaciones de ERP se convirtieron en catalizadores y facilitadores de muchas actividades de reingeniería corporativa [219].
- Los ERP estaban ya adaptados para que no se vieran afectados por el problema del año 2000 (Y2K) [219, 222].
- La frontera de ERP fue presionada por proveedores (con organizaciones de investigación y desarrollo) agresivos y poderosos [219].

Hoy en día, según [223] hay una transformación de ERP hacia ERP II. GartnerGroup [224] define a *ERP II* como una transformación de ERP en una nueva generación de sistemas empresariales. Una estrategia empresarial y un conjunto aplicaciones empresariales de dominio específico que crea valor en clientes y accionistas al permitir la optimización del funcionamiento de las empresas y de las relaciones existentes inter-empresas.

ERP II tiene como principal componente el soporte a los procesos de e-Business y la colaboración en la cadena de suministro. El término *e-Business* hace particular referencia a la adopción de Internet para fomentar las nuevas herramientas en el fortalecimiento de los procesos de negocio y en la aceleración del objetivo de integración de la cadena de suministro.

ERP II tiene como objetivo la gestión de la cadena de suministro en su totalidad integrando y coordinando actividades a través de la frontera de la organización.

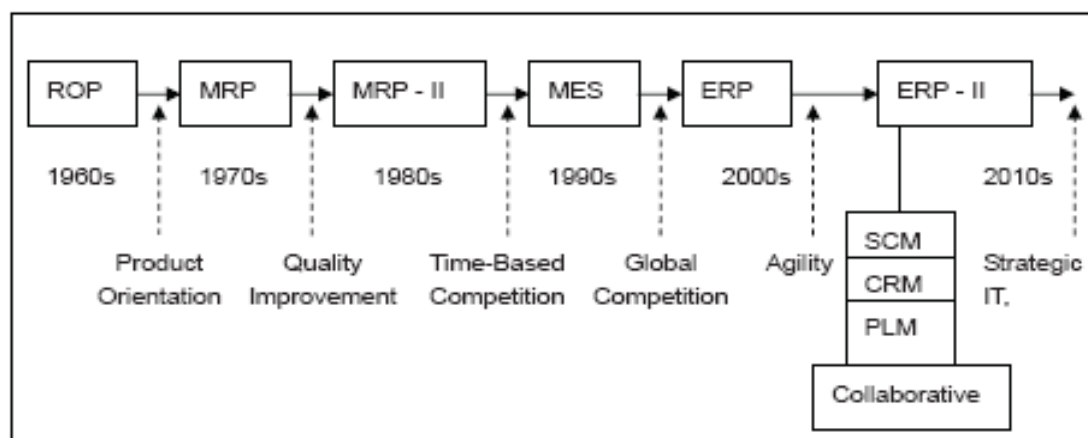
En los últimos años sigue la tendencia de crecimiento de los ERP. Según declaraciones de Bo Lykkegaard, encargado de la investigación europea de la empresa en IDC. [225] “2006

*parece ser el año del retorno de los ERP en Europa occidental*”. No se equivocaba, ya que [226] destaca que el 2006 fue un año con grandes ganancias para el mercado de los ERP, con unos ingresos que llegaron a 28 billones de dólares.

En el 2007, el mercado europeo de ERP se ha beneficiado del reemplazo de ERP implantado a finales de los noventa, así como de una mejora en las condiciones de negocio en Europa en general. Para este mismo año se preveía que el mercado de los ERP en Europa, Oriente Medio y África generaría 6.7 mil millones euros en rédito total del software. Esto representa el crecimiento de 5.7% comparados con 2006. En el estudio realizado por [227] se preveía que para ese mismo año que el 17% de las empresas europeas planificaban una actualización en sus ERP.

Según [228], en España, el 61.9% del negocio total relacionado con los ERP (641,85 millones de Euros) corresponde a servicios con unas ventas totales de 397.16 millones de Euros, de los cuales el 29.2% corresponde a software, con unas ventas de 187.13 millones de Euros.

Por su parte, [226] estima que para el 2008 las ganancias estimadas serán de 35.8 Billones de dólares, 39.4 billones de dólares, para el 2009, 43.4 billones de dólares para el 2010.



**Figura 2-O: Evolución de los ERP**

En la figura 2-O [229] se muestra la evolución del mercado de los ERP desde su surgimiento hasta nuestros días. Como se puede observar, continúa creciendo, generando impacto en las empresas alrededor del mundo. Actualmente, es considerado como uno de los mercados con un crecimiento mas acelerando en la industria del software. Por lo que se espera, un continuo crecimiento en los próximos años y mucho trabajo futuro.

## 2.6.2 Definición y Características

El concepto de ERP se puede ver desde una gran cantidad de perspectivas [221], para el desarrollo de esta tesis hemos considerado la definición de ERP aportada por [230] como la mas adecuada:

*Un ERP es un paquete software (packaged software), generalmente compuesto por múltiples módulos, que ofrece soluciones integradas diseñadas para dar soporte a múltiples procesos de negocio (recursos humanos, ventas, finanzas, producción, etc.), proporcionando una integración de los datos de la organización con los procesos de negocio.*

Un ERP tiene un diseño genérico para poder abarcar gran diversidad de empresas de tipos distintos y por tanto con características muy variadas [221, 231] y por tanto refleja la forma en la que en general operan las empresas [219]. Por esta razón necesitan una configuración personalizada para cada empresa partiendo de una parametrización inicial del ERP (la parte que es común a todas las empresas), definiendo valores asignados a los parámetros de control del ERP durante la implementación, lo que determina las operaciones y procesos exactos que soportara el ERP en una empresa específica [232, 233].

Por tanto podemos decir que las características que distinguen a un ERP de cualquier otro software empresarial son que deben de ser sistemas [234]:

- **Integrales.** Deben permitir controlar los diferentes procesos de la organización, entendiendo que todos los departamentos de una empresa se relacionan entre si. Por lo que:
  - Poseen una base de datos centralizada controlando así la redundancia de los datos [221].
  - Los datos se ingresan sólo una vez y deben ser consistentes, completos y comunes.
  - Están diseñados para trabajar en varios países, por lo cual pueden manejar requisitos específicos a diferentes regiones [221].
- **Modulares.** Una empresa está formada por un conjunto de departamentos que se encuentran interrelacionados por la información que comparten y que se genera a partir de sus procesos. Los ERP, tanto económica como técnicamente, deben tener dividida su funcionalidad en módulos [221, 230], los cuales pueden instalarse de acuerdo con los requisitos del cliente. Por lo que:
  - Soportan una alta funcionalidad de negocio, brindando funciones de negocio, especialmente producción, logística, manejo de materiales, ventas, distribución, financieras, contables, plantación estratégica, gestión de calidad y compras [221].
  - Sus módulos interactúan entre sí consolidando todas las operaciones (llamado *funciones cruzadas*), por lo que en muchas ocasiones el usuario no se da cuenta en qué modulo esta trabajando [221].

- **Adaptables.** Deben estar creados para adaptarse a la idiosincrasia de cada empresa (debido a que son un paquete de software estándar [221]). Esto se logra por medio de la configuración o parametrización de los procesos de acuerdo con los requisitos específicos de la empresa. La parametrización es el valor añadido fundamental que se debe hacer con cualquier ERP para adaptarlo a las necesidades concretas de cada empresa. Por lo que:
  - Imponen su propia lógica a la estrategia, organización y cultura de la empresa [219, 221]. Las empresas que los implantan suelen tener que modificar alguno de sus procesos para alinearlos con los del ERP. Este proceso se conoce como *Reingeniería de Procesos* [235].
  - La tendencia actual es que ofrezcan también funcionalidades especializadas para determinadas empresas [221] (hospitales, universidades, gobierno, etc.). Es lo que se denomina *versiones sectoriales o aplicaciones sectoriales* especialmente indicadas o preparadas para determinados procesos de negocio de un sector.

Dichas características inducen a que las empresas tengan una serie de ventajas si implementan un ERP, respecto a una solución más tradicional:

- Optimización de los procesos empresariales.
- Acceso a información confiable, precisa y oportuna.
- Mejoras en la comunicación entre las áreas de producción.
- Reducción en la duplicidad de información.
- Mayor eficiencia en la integración de los procesos comerciales.
- Reducción de tiempos en los costos de los procesos.
- Mayor información para la toma de decisiones.
- Extensión de las mejores prácticas y difusión de conocimiento a lo largo de la organización.

Claramente las ventajas de la implementación de un ERP son enormes y existe una gran cantidad de casos de éxitos registrados. Esto provoca en las empresas la prisa por implementar un ERP sin tener un claro entendimiento de los procesos de negocio y por tanto de sus requisitos [219]. Este desconocimiento trae como resultado una ineficaz selección del ERP y por tanto del alineamiento de los requisitos de la empresa con los del ERP. Trayendo como resultado una gran cantidad de casos de fracaso en la implementación.

### 2.6.3 Módulos

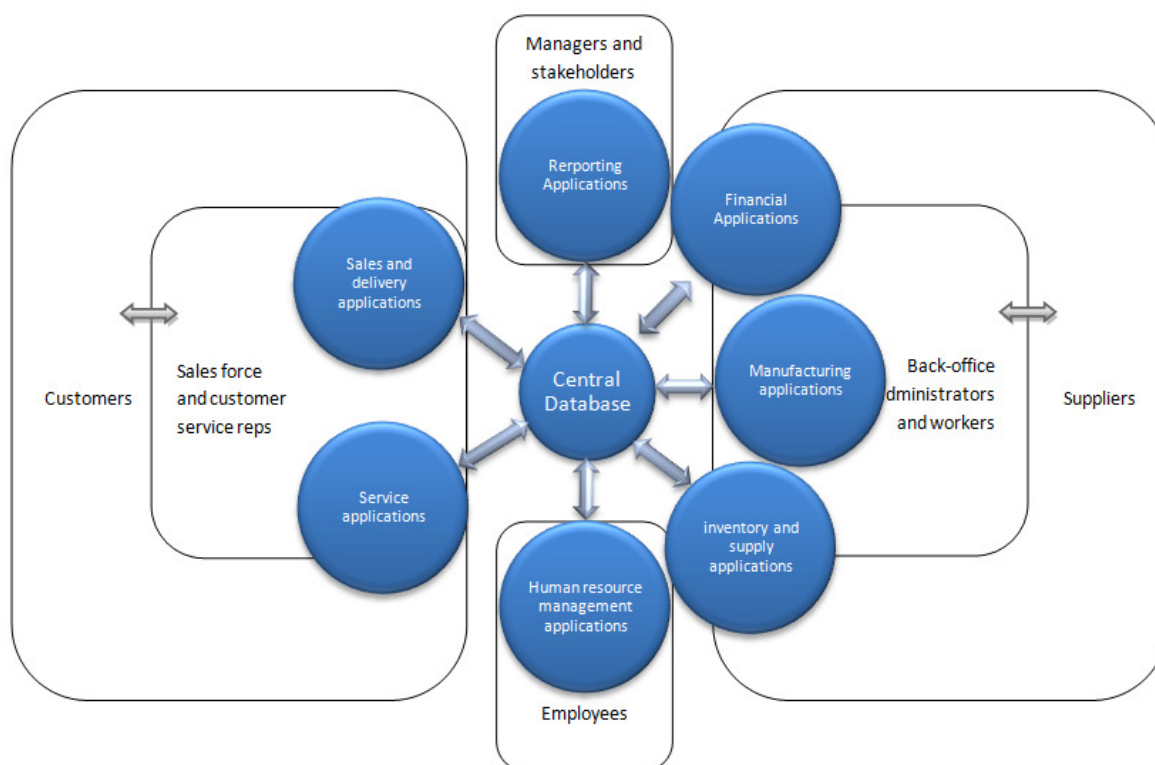
Un ERP esta compuesto por módulos, esto permite a una empresa implementar algunas funciones y otras no [219]. En ocasiones, una empresa simplemente no necesita un módulo (por ejemplo una empresa de servicios no necesita un modulo de manufactura). En otros casos, la empresa ya cuenta con el sistema que realiza las funcionalidades de algún modulo en particular. En general, mientras más módulos se implementen, mayores serán los beneficios de integración, pero también serán mayores los costes, los riesgos y los cambios. Los módulos permiten personalizar los ERP en cierto grado.



Estos módulos se integran permitiendo compartir y transferir información libremente, centralizando toda la información en una única base de datos accesible a todos los módulos [236]. La figura 2-P muestra esta integración [219].

Catalogándose principalmente en tres tipos [237]:

- Básicos: Suelen ser módulos obligatorios a adquirir.
- Opcionales: Son los módulos no obligatorios y permiten añadir nuevas funcionalidades al paquete ERP.
- Verticales: Son módulos opcionales diseñados específicamente para resolver las funcionalidades de un sector específico como la administración pública, hospitales, la banca, etc.



**Figura 2-P: Anatomía de un ERP**

El número, tipo, nombre y funcionalidad de los módulos que provee un ERP puede variar según el proveedor. Ésta variabilidad en los módulos se presenta como consecuencia del crecimiento en la capacidad de los ordenadores y de la propagación del uso de Internet. Lo que provoca que los proveedores de ERP y los clientes rediseñen los ERP continuamente, rompiendo con esto la barrera de la propiedad y la personalización de requisitos particulares. Este rediseño provoca un proceso interminable de reingeniería y desarrollo trayendo productos y soluciones nuevas al mercado de ERP [238]. Así, cada fabricante tiene sus propias líneas de productos y soluciones, sin embargo, a pesar de las diferencias, estos paquetes tienen módulos y funcionalidades similares [239].

Con el fin de identificar los módulos básicos de un ERP se han analizado distintas fuentes de información, considerado:

- Fuentes de información accesibles sobre los ERP líderes en el mercado. Según [226] ORACLE y SAP son los líderes en el mercado y continuaran siéndolo según lo previsto para los próximos años.
- Los principales proveedores de ERP que han sido seleccionados o vistos durante los procesos de selección de ERP realizados por una empresa consultora dedicada entre otros a la selección de ERP en Europa y que hemos tomado como base son:
  - Alizee Soft Orchestra
  - Microsoft dynamics Axapta
  - Microsoft dynamics Navision
  - PMIX
  - ORCHESTRA
  - IFS

Para organizar la información obtenida a partir de las fuentes, mediante la tabla 2-E se ha realizado una comparativa de las funcionalidades ofrecidas en los módulos disponibles por cada uno de los ERP. La información obtenida se basa en un análisis del mercado, considerando las especificaciones ofrecidas por los proveedores de los ERP, y no considera ningún modelo de negocio en particular. Una definición de cada una de las funcionalidades se puede ver en el anexo 1

	JDEdwards (Oracle)	ORACLE	SAP	Axapta (Microsoft Dynamics AX)	People Soft (Oracle)	Navision (Microsoft Nav)	ORCHESTRA	pmiX	IFS
Finanzas	✓	✓	✓	✓	✓	✓	✓	✓	✓
Capital Humano	✓	✓	✓	✓	✓	✓	✗	✗	✓
Proyectos	✓	✓	✓	✓	✓	✓	✓	✓	✓
Producción	✓	✓	✓	✓	✗	✓	✓	✓	✓
Compras	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ventas	✓	✓	✓	✓	✓	✓	✓	✓	✗
Análisis Corporativo	✗	✓	✓	✓	✓	✓	✓	✗	✓
Mantenimiento	✓	✓	✓	✓	✓	✓	✗	✓	✓
Marketing	✗	✓	✗	✓	✗	✓	✗	✓	✓
Servicios	✗	✓	✓	✓	✓	✓	✗	✓	✓
Logística e inventario	✓	✓	✓	✓	✗	✓	✓	✓	✓
Cuentas por pagar	✓	✓	✓	✓	✓	✓	✓	✗	✓
Cuentas por cobrar	✓	✓	✓	✓	✓	✓	✓	✗	✓
Control en piso	✗	✓	✓	✓	✗	✗	✓	✗	✓
Aplicaciones cruzadas	✓	✓	✓	✓	✓	✓	✗	✗	✗
Gestión del control de calidad	✗	✗	✓	✗	✗	✗	✗	✓	✓

**Tabla 2-E: Comparativa entre módulos ERP**

Considerando esta tabla comparativa y tomando en cuenta el análisis de mercado de los ERP realizado por [159] así como los libros de requisitos proporcionados por la empresa consultora antes mencionada, se han identificado como módulos básicos de un ERP los siguientes:

**Finanzas:** Da una visión completa de la situación financiera y permite controlar todas las operaciones de la organización. Incluyendo la toma de decisiones basadas en información financiera en tiempo real. Aumenta la transparencia y velocidad de los reportes e informes financieros permitiendo afrontar las necesidades corporativas día a día. Incluye entre otras: gestión y análisis financiero, gestión contable, activos fijos y tesorería.

**Ventas:** Está enfocado en acelerar el ciclo de ventas desde la administración de ventas hasta la efectividad de las mismas. Incluye entre otras la gestión de órdenes de venta, marketing, servicio post-venta y mantenimiento.

**Compras:** Permite gestionar las posibles complicaciones relacionadas con el proceso de compra tanto de bienes como de servicios. Incluye entre otras: dirigir la consecución de las operaciones, colaboración con proveedores, abastecimiento estratégico, subcontrataciones y gestión de requisitos y órdenes de compras.

**Producción:** Ayuda a optimizar la capacidad de producción en todas las fases del proceso de fabricación desde la materia prima hasta el producto final. Permite la planificación, la ejecución, el control y el análisis de distintos tipos de fabricación. Incluye entre otros: La gestión del ciclo de vida del producto, gestión del proceso de producción, gestión del tiempo y entregas.

**Recursos Humanos:** Permite el manejo eficiente de la fuerza de trabajo. Proporciona las herramientas necesarias para alinear a los trabajadores con los objetivos de la empresa. Incluye entre otras: la gestión de los recursos humanos, pagos de nóminas, reclutamiento, gestión del tiempo y tareas, formación y necesidades analíticas.

**Logística:** Controla la ejecución total del proceso desde la gestión del almacén hasta el transporte y vuelta al almacén. Optimiza el flujo de materiales a través del canal de suministro con una continua gestión. Incluye entre otras: la gestión del inventario, la gestión logística y la gestión de transportación.

**Análisis:** Brinda informes y análisis diseñados para entregar información procesable, en tiempo real y de forma precisa tanto a ejecutivos como a directivos y empleados con responsabilidad en la toma de decisión. Incluye entre otras: Gestión estratégica de la organización, análisis financiero, operativo y de mano de obra.

**Mantenimiento:** Proporciona planes de mantenimiento proactivos gestionando el servicio de mantenimiento ayudando a mantener el gasto de mantenimiento en un mínimo. Ayuda a cubrir todos los aspectos del ciclo del servicio de mantenimiento, los presupuestos, la planificación del mantenimiento y el seguimiento de las fichas de servicio. Incluye entre otras: la gestión de la garantía y reparación, ordenes de servicio y las etapas del proceso.

**Proyectos:** Mejora los proyectos de la empresa y gestiona los recursos de la misma racionalizando el ciclo de vida completo de dichos proyectos desde la venta hasta la entrega e integrándolos con las operaciones de la empresa. Incluye entre otros: la gestión de los recursos, del tiempo y de las necesidades de desarrollo tanto de la organización como de sus empleados, calcula costos, precios y presupuestos.

**Aplicaciones cruzadas.** Describe funcionalidades comunes. No es relativo a un modulo específico sino a todos los módulos. Incluye entre otros: personalización de la interfaz de usuario, alertas, reportes, registros y procesamiento en batch.

#### **2.6.4 Relación entre los ERP y otros sistemas de gestión.**

El propósito de colaboración entre módulos se extiende a nivel de otros sistemas de gestión que permiten la entrada y salida de información. Esta relación, permite mejorar y ampliar las funciones centrales de los ERP, proporcionando apoyo a las decisiones para la gestión de las relaciones y las empresas. Estos componentes no son necesariamente sincronizados de forma directa con la base de datos integrada [223].

A continuación se describen los dominios a los que pertenecen los componentes OTS que pueden depender o de los que pueden depender los ERP [223]:

- **Supply Chain Management (SCM):** Los componentes OTS correspondientes a este dominio soportan todo lo relacionado con la planificación y la producción de bienes. Por ejemplo, un SCM puede proporcionar información sobre los productos que se van a producir, la adquisición de piezas y los plazos de entrega.
- **Customer Relationship Management (CRM):** Los componentes OTS correspondientes a este dominio facilitan un amplio conjunto de funcionalidades, entre las que destacan el proceso de identificación del cliente, y la gestión de servicio al cliente.
- **Supplier Relationship Management (SRM):** Los componentes OTS correspondientes a este dominio son análogos al CRM pero en lo relacionado con los proveedores y compras de la organización. Los SRM permiten a las empresas gestionar sus relaciones con los proveedores en la totalidad de su ciclo de vida.
- **Product Lifecycle Management (PLM):** Los componentes OTS correspondientes a este dominio, y habitualmente incluyendo un componente OTS del dominio Product Data Management (PDM), permiten a las empresas llevar los productos innovadores y rentables al mercado eficazmente, especialmente en el ámbito de e-business. Los PLM permiten aprovechar el proceso de innovación de las empresas mediante la gestión eficaz de la definición completa del ciclo de vida del producto.
- **Employee Lifecycle Management (ELM):** Los componentes OTS correspondientes a este dominio permiten la integración de todos los aspectos de la información de un empleado desde su contratación hasta su retiro de la compañía. Los ELM permiten la gestión eficaz del portafolio de competencias.
- **Corporate Performance Management (CPM):** Los componentes OTS correspondientes a este dominio son usados para supervisar y gestionar el rendimiento empresarial. Los CPM son componentes software que proporcionan a la administración de la organización una perspectiva general sobre la empresa, permitiendo definir y gestionar metodologías, métricas, procesos y sistemas utilizados por la empresa.

En la figura 2-Q se muestra la relación de estos dominios con el dominio de los ERP reflejando su relación con e-Business lo que permite la colaboración y la relación con agentes externos [240].

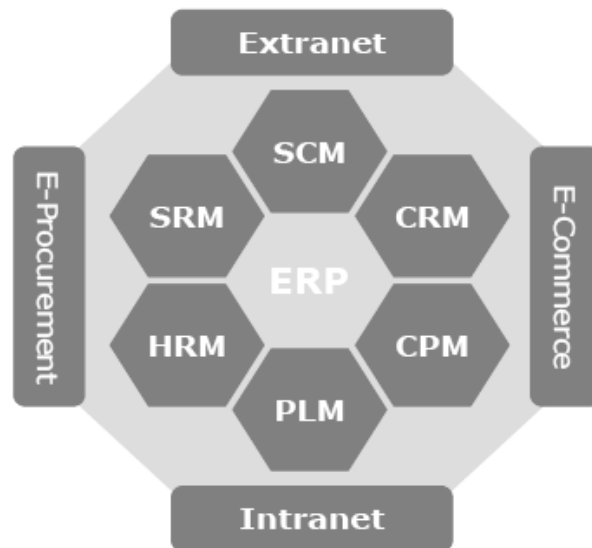


Figura 2-Q: Relación de ERP con otros sistemas

## 2.6.5 Selección de ERP

La importancia del impacto de los ERP en los procesos cotidianos de la organización, y la inversión que la empresa debe hacer en términos económicos, hacen que el proceso de selección de la herramienta sea un tema delicado. Se debe tener en cuenta que no es una tarea que se haga frecuentemente y que se espera un determinado retorno de la inversión en términos monetarios y de tiempo de uso.

En la sección 2.6.5.1 se enmarca la selección de ERP en el ciclo de vida global de un ERP desde que se decide su adquisición hasta su retiro. En la sección 2.6.5.2 se presentan los casos de éxito y de fracaso de los procesos de selección e implantación de ERP. En la sección 2.6.5.3 se describe la importancia de la selección del ERP dentro del ciclo de vida. En la sección 2.6.5.4 se describe el trabajo que nos consta que se ha hecho sobre selección de ERP.

### 2.6.5.1 La selección de los ERP, ciclo de vida

El ciclo de vida de un ERP (ver figura 2-R) es similar al de otros componentes OTS sin embargo, existen algunas diferencias debido a la importancia que toma la parte social y organizacional en este tipo de componentes. Consiste en las siguientes etapas [230]:

- **Decisión de adopción:** En esta fase se deben cuestionar la necesidad de un nuevo ERP, mientras se selecciona la información general del sistema que mejor se puede adaptar a los desafíos y estrategia del negocio. Incluye la definición de los requisitos del sistema, sus objetivos y beneficios, y el análisis del impacto de adopción del ERP a nivel de negocio y organizacional.
- **Adquisición:** Consiste en la selección del producto que mejor encaje con los requisitos de la organización. También suele seleccionarse una consultoría que ayude en las siguientes fases del ciclo de vida del ERP, especialmente la fase de implementación. Factores como el precio, entrenamiento y mantenimiento son

analizados y los acuerdos contractuales son definidos. En esta fase también es necesario hacer un análisis del retorno de la inversión del producto seleccionado.

- **Implementación:** Esta fase consiste en la personalización, parametrización y adopción del ERP adquirido de acuerdo a las necesidades de la organización. Usualmente esta tarea es hecha con la ayuda de consultores quienes proveen la metodología para la implementación, el know-how y el entrenamiento.
- **Uso y mantenimiento:** Esta fase consiste en el uso del producto. En esta fase se debe estar enterado de los aspectos relacionados con la funcionalidad, la utilidad y la suficiencia de los procesos organizacionales y de negocio. Una vez que el sistema se ponga en ejecución, se debe realizar mantenimiento, porque el malfuncionamiento tiene que ser corregido. También se debe conocer los requisitos de optimización especial y las mejoras generales al sistema deben hacerse.
- **Evolución:** Esta fase corresponde a la integración del ERP con otros softwares de la organización, como pueden ser, SCM, CRM, workflow; y también expandiendo las fronteras externas con otras organizaciones.
- **Retiro:** Esta fase se aplica cuando aparece una nueva tecnología que pueda suplir al ERP o, en caso que, el ERP sea insuficiente para lo que el negocio necesita. Los encargados deciden sustituir el ERP con el otro que sea más adecuado para las necesidades de la organización en ese momento.



Figura 2-R: Ciclo de vida de los ERP

### 2.6.5.2 Casos de éxito y fracaso en proyectos de selección de ERP

Como hemos comentado anteriormente, a pesar de la gran cantidad de casos de éxito en la selección e implementación de los ERP, existen también una gran cantidad de fracasos y problemas durante todo el ciclo de vida del ERP [219]. A continuación destacamos algunos ejemplos de estas dos situaciones:

- **Casos de éxito:**
  - Mahindra and Mahindra redujo el 30% de su inventario;
  - Marico incrementó su utilidad un 1.5%;
  - ABB incremento su eficacia operacional del 15% al 20%;
  - BPCL ahorró 7.5 millones de euros;
  - L and T recobró su inversión en dos años [241];
  - Colgate, disminuyó el costo de su inventario;
  - Ericsson redujo el tiempo de procesamiento en ordenes de venta, programación de la producción y ordenes de compra [242].
- **Casos de fracaso:**
  - FoxMeyer Drug argumentó que la implementación de un ERP lo llevo a la bancarrota,
  - Mobil Europe gastó cientos de millones de dólares en un ERP solo para abandonarlo cuando su socio se opuso,
  - Dell Computer encontró que el ERP no encajaba con su nuevo modelo descentralizado.
  - Applied Materials abandonó el ERP cuando se vió ahogado por los cambios de organización,
  - Dow Chemical gastó 7 años y medio billones de dólares para finalmente dejarlo y empezar con una versión cliente-servidor.

De aquí la necesidad de seleccionar el ERP que se adapte mejor a los requisitos de la organización, para ello se requiere de un arduo trabajo que conlleva a problemas organizacionales. Ciertamente el no hacer un buen trabajo de selección del sistema ERP puede poner a la compañía en graves problemas a tal grado que puede llevar a la quiebra a empresas pequeñas o medianas.

### **2.6.5.3 Importancia de la selección del ERP dentro del ciclo de vida**

De acuerdo con [243] La etapa de selección del ciclo de vida del ERP es una de las más difíciles y de las más críticas, por lo que es necesario poner mayor atención a esta etapa. Por lo que se requiere de un proceso de selección que se distingue por [244]:

- Definir claramente una necesidad que podría ser satisfecha mediante un producto ERP o servicio relacionado.
- Permitir encontrar los productos y servicios adecuados en el mercado que ayudan a cumplir tal necesidad.
- Facilitar el establecimiento de criterios apropiados para la evaluación de ERPs.
- Ayudar a evaluar productos y servicios a la luz de criterios establecidos.
- Permitir seleccionar el producto y servicio más adecuado, o su combinación.
- Ayudar a negociar el contrato final con el vendedor del producto y el proveedor del servicio.

La selección de un ERP es un proceso organizacional, de aprendizaje y un proceso social debido a [245]:

- **Organizacional.** Los procesos de selección de ERP se pueden considerar procesos sociales complejos debido a, por una parte, la participación de personas en diversos centros de responsabilidad organizacionales (departamentos, procesos, proyectos). Y, por otra parte, al tiempo que se consume en reuniones y toma de decisiones, con la consiguiente inversión y uso de recursos humanos.
- **De aprendizaje.** El proceso de selección de ERP es un proceso de aprendizaje y conocimiento de la propia organización y sobre el funcionamiento futuro deseado.
- **Social.** El proceso de cambio social en el ámbito organizacional comienza a gestarse en el propio proceso de selección del ERP conforme la estructura organizacional y de negocio, presente y/o deseada, comienza a encontrar referentes de soporte más o menos automatizado en los diversos módulos del ERP. Con esto, las personas, como sistemas cognitivos ven nuevas posibilidades y, como sistemas humanos, comienzan a concebirse como artefactos o piezas organizacionales engranados dentro de una nueva tecnología llamada ERP. Todo lo anterior convierte al proceso de selección de un ERP en un proceso socio-técnico singular y particular dentro del cual deben negociarse continuamente los deseos humanos con las posibilidades tecnológicas y las necesidades de empresa.

Por lo que es necesario estudiarlo y realizar métodos que faciliten la selección de un ERP tomando en cuenta las connotaciones del proceso.

#### **2.6.5.4 Métodos para la selección de ERP**

A pesar de la importancia de la selección de ERP y de los procesos que implica, los estudios en el ámbito de los ERP se ha enfocado principalmente a la implementación, y la literatura relacionada con la selección de ERP es escasa. En la tabla 2-F se describen algunos de ellos.

Del análisis de estos métodos podemos observar que para poder realizar una correcta selección de un ERP, es crucialmente necesario seleccionar aquellos que contengan la mayor parte de los requisitos de la empresa dentro de las funcionalidades que proporciona el ERP. Desafortunadamente, uno de los problemas que se encuentra comúnmente es el alineamiento de estos requisitos para la adaptación del ERP [235, 246]. Por esto, es necesario contribuir con soluciones prácticas al problema del entendimiento y la expresión de los requisitos para obtener un buen análisis y por consiguiente una correcta selección de ERP.



Perspectiva	Publicaciones	Descripción
<p><i>Ingeniería de requisitos:</i> Se centran en la ingeniería de requisitos definiendo atributos medibles. La selección se basa en la puntuación de que tanto los ERP se adecuan a los requisitos de la organización.</p>	SHERPA [244]	Va desde la búsqueda del candidato hasta la firma del contrato, utiliza investigación acción para comprender el problema en base a la experiencia, hace hincapié en requisitos de estrategia de negocio, construye tablas de criterios de selección para cada fase, la evaluación se va refinando y enriqueciendo filtrando candidatos. Utiliza el lenguaje No-fun para formalizar.
	MSSE [247]	Toma como base la metodología SHERPA y agrega para la selección encuestas clasificadas por módulos ERP. Se basa para una primera evaluación en los requisitos funcionales y en global se basa en los requisitos en general, proponiendo un proceso para evaluar y seleccionar candidatos de consultoría.
	AHP method for ERP selection [248]	Ayuda a la construcción sistemática de los objetivos para la selección del ERP cubriendo las estrategias y objetivos de la organización. Identifica las características de los ERP y filtra a los proveedores que no cumplen con los requisitos que son parte de la estructura de objetivos de la organización. La evaluación la realiza utilizando el método AHP.
	FAHP [249]	En este modelo hay 32 criterios tamizados a partir de los requisitos de productos y los requisitos referentes al aspecto de gestión. Para evaluar las alternativas del sistema ERP este modelo considera que las características de calidad de software basándose en el ISO 9126.
	InteliTeam [250]	Se basa en el paradigma de Courtney's DSS. Presenta un marco colaborativo basado en web, con un enfoque de múltiples perspectivas para la gestión del conocimiento y la toma de decisiones en un problema organizacional. Presentando una selección de ERP con Inteliteam para una empresa hipotética para ilustrar la eficacia del marco propuesto
	MERPAP [251]	Presenta un modelo basado en el resultado de casos de estudio, dando seis distintos procesos iterativos. Se centra en la adquisición del ERP mediante un estudio de selección y evaluación de los proveedores y del sistema. Para la evaluación considera al proveedor, la parte funcional y la técnica.
	A framework for evaluating ERP projects [252]	Se basa en la comprensión del mapa completo de las características de los ERP siguiendo 10 criterios propuestos. Incorpora el aprendizaje participativo y el proceso de toma de decisiones basándose en Nominal Group Technique (NGT) y adapta la metodología de evaluación de AHP. (Aunque algunos de los criterios de selección se basan en costo-beneficio no son los únicos y se da igual importancia que el resto de requisitos).
	The 2-tuple linguistic model [253]	Se basa en el procesamiento lingüístico de información 2-tuple para hacer frente al problema de selección de un ERP. Se propone un algoritmo basado en el grado de similitud para agregar la información objetiva sobre

		los ERP de algunas organizaciones profesionales externas, que podrían ser expresadas por diferentes conjuntos de expresión lingüística. Los índices de consistencia e inconsistencia se definen considerando la información obtenida de entrevistas internas con los proveedores de ERP y, entonces, se establece un modelo de programación lineal para optar por el ERP más adecuado.
<i>Retorno de la inversión:</i> La selección se basa en el análisis de costo-beneficio de la implementación del ERP.	Ex-ante ERP evaluation framework [254]	Se centra en aclarar los objetivos de la organización, considerándolos como lo más importante. Reconoce que el vendedor, el producto y los servicios deben ser evaluados pero la decisión final debe basarse en la cantidad de cambio que requiere la organización para la implementación del ERP y el retorno de la inversión.
<i>Decision making:</i> Se basa en la influencia de los stakeholders en el proceso de selección de un ERP.	Decision Making in the Evaluation, Selection and Implementation of ERP Systems [255]	Este estudio trata de mapear 6 modelos de la toma de decisiones en ERP, proponiendo el conocimiento de estos modelos para una mejor planificación de cada fase del ciclo de vida del ERP
	Towards a Model for Investigating Non-Decision Making in ERP Communities [256]	Implementa el proceso de Decisión Making a la selección e implementación de paquetes ERP e introduce el concepto de Non-Decision-Making
	Investigating Non-Decision Making during an ERP Software Selection Process [257]	Se basa en el marco teórico propuesto por Sammon y Adam para comprender la dinámica de selección de los ERP destacando la no-toma de decisiones (NDN) en el ERP.

**Tabla 2-F: Métodos para la selección de ERP según perspectiva**

### 2.6.6 Contribución del grupo GESSI a la selección de ERP

En [258] se expone un conjunto de declaraciones relativas a las adquisiciones ERP, proponiendo que el proceso de adquisición debe estar bien definido y ser sistemático.

Debido a la falta de métodos particulares para la adquisición de componentes OTS, se crea SHERPA [259]. SHERPA es un nuevo método para la adquisición de componentes ERP basado en descripciones de lenguaje natural del dominio de aplicación, las necesidades de los usuarios y el candidato ERP. Posteriormente se propone en [19] la aplicación de un lenguaje formal en SHERPA para conseguir resultados mas fiables y comprensibles.

En muchas ocasiones se necesita seleccionar más de un componente OTS. Por ello en [260] se propone un modelo combinado de selección de componentes basado en la distinción de niveles mundial y local.

[261] Describe un caso de implementación de un ERP en Portugal. Se enfoca en la identificación de factores organizacionales detectando problemas de implementación relacionas con estos.

### **3 Construcción de un catálogo de patrones de requisitos funcionales para ERP**

Tal como hemos visto en el estado del arte, un componente ERP es una solución genérica, sin embargo cada organización es única y existen muchas opciones de ERP en el mercado en donde cada una de ellas tiene características especiales. Cada organización debe elegir la opción más adecuada para satisfacer sus necesidades.

Como hemos dicho anteriormente el proceso de selección de un ERP no es una tarea fácil, y por tanto requiere de especial atención. En concreto, para realizar dicho proceso, se requiere alinear el conjunto de requisitos que se han establecido para el ERP a adquirir, con los requisitos organizacionales. Con el fin de facilitar este proceso de alineamiento y ayudar a identificar las necesidades de la organización, los requisitos deben ser expresados de forma que puedan ser fácilmente identificados para su posterior uso. Por lo que podemos decir que los requisitos sirven como artefactos de reutilización [235].

Por lo general, los requisitos son expresados en lenguaje natural y esto conlleva problemas que son bien conocidos (principio 53 [262]). Con el fin de expresar los requisitos de forma clara y que puedan ser reutilizables hacemos esta propuesta de procedimiento para la construcción de patrones de requisitos. Este procedimiento, tiene como objetivo la obtención de un catálogo de patrones que puedan ser fácilmente generalizados, identificados, medidos y categorizados.

En este capítulo exploraremos el método propuesto para la construcción de un catálogo de patrones de requisitos funcionales para ERP. El capítulo se divide en: En la sección 3.1 se define que entendemos por patrón de requisitos funcionales, se describe la notación, estructura y las plantillas propuestas, así como la clasificación del patrón. En la sección 3.2 se describen las consideraciones previas para el mejor entendimiento del método propuesto, se justifica la importancia de la existencia de un catálogo de patrones de requisitos funcionales, y se describe el contexto para la construcción de patrones de requisitos funcionales de ERP, En la sección 3.3 se presenta el método propuesto de acuerdo a los pasos que hemos seguido para la construcción de los patrones. En la sección 3.4 se presentan un diagrama UML para el mejor entendimiento de la creación de patrones. En la sección 3.5 se realiza un ejemplo siguiendo el método propuesto en la sección 3.3. En la sección 3.6 se describen las aplicaciones que proponemos.

### 3.1 Definición

Como hemos visto en el estado del arte los patrones han sido usados en distintos ámbitos de la ingeniería del software, creados en primera instancia para resolver problemas identificados durante el diseño de paquetes software [10]. En este caso nos centramos en la creación de patrones de requisitos funcionales. Definiéndolo como:

*Un patrón de requisitos funcionales es un artefacto reutilizable que se encuentra durante las distintas actividades del proceso de ingeniería de requisitos y que representa de forma estructurada y abstracta una o varias necesidades funcionales de una organización*

La notación usada para su construcción es la siguiente:

*Objetivo del patrón <debe> propiedad a cumplir {variables}*

Donde:

- El objetivo del patrón describe el objetivo que debe alcanzar el patrón.
- La propiedad a cumplir describe aquello que es necesario cumplir para alcanzar el objetivo.
- Los parámetros describen la parte del patrón que dependerá de cada proyecto cuando se use. No siempre aparecen en los patrones.

Por ejemplo: “El cálculo del precio de venta debe realizarse en dependencia de los parámetros SP”.

**Estructura del patrón.** El patrón se compone de dos partes:

- **Parte fija.** Corresponde propiamente a la definición del patrón tomando en cuenta la estructura descrita anteriormente. Es una frase que expresa el propio objetivo es decir el propósito general del patrón y sus propiedades. Esta parte será reproducida de la misma forma cuando se aplique el patrón en un proyecto concreto.
- **Extensión(es).** Una o varias extensiones de la parte general. Son opcionales, no serán necesarias en cualquier caso donde se aplica la parte general. Podían también ser incluidas en el catálogo como diversos patrones. Sin embargo pensamos que es mejor tenerlas juntas puesto que el conjunto contribuye a expresar un mismo objetivo para el componente que se seleccionará. Por lo que podemos decir que las extensiones nos ayudan a identificar subconjuntos de requisitos funcionales.

En la figura 3-A se ejemplifica la definición de un patrón de requisitos funcionales para ERP. La parte fija (el patrón) corresponde al punto 1 y la extensión del patrón está ejemplificada en el punto 1.1 mostrando su dependencia con la parte fija del mismo.

1	Parte fija del patrón	El cálculo del precio de venta debe realizarse en función de los parámetros SP
1.1	Extensión	La fórmula de cálculo debe depender del cliente que compra el del cliente que compra el producto

**Figura 3-A: Componentes de un patrón**

Las plantillas de requisitos ayudan a expresar e identificar los requisitos fácilmente. De esta forma y para formalizar un patrón de requisitos funcionales, se ha creado una plantilla que muestra su estructura. Para ello se ha tomado como base la propuesta en [105, 263], y los principios de desarrollo de software descritos en [262] de forma que se define:

- **Identificador y nombre del patrón:** Cada patrón funcional es único y ha de ser fácilmente identificado. En este caso mediante un número y un nombre descriptivo (principio 52 [262]).
- **Módulo ERP.** Indica el módulo o módulos del ERP a los que corresponde el patrón.
- **Palabras clave.** Contiene las palabras clave básicas para su rápida ubicación.
- **Versión:** Permite almacenar las versiones de los requisitos para conocer su objetivo y versión originales pudiendo así modificarlo de forma segura (principio 43 [262]).
- **Autor:** Contiene el nombre y la organización del autor de la versión.
- **Fuente:** Contiene la fuente de la versión para poder ubicar el origen.
- **Objetivo del patrón:** Describe el objetivo que debe alcanzar el patrón, ayudando a quien realice la elicitación de requisitos en la selección de patrones, en la validación de nuevos requisitos, y en la generación de nuevas ideas sobre las relaciones entre patrones.
- **Descripción:** Contiene un breve resumen que describe el patrón en general. Sirve como introducción al patrón y puede incluir referencias a los patrones que están relacionados con el.

**Parte fija del patrón:** Tomando en cuenta la notación descrita anteriormente (principio 52 [262]), describe el objetivo del patrón y la propiedad a cumplir, más un conjunto de parámetros (si así se requiere) el patrón y la forma de medirlo denominados:

- **Nombre del parámetro:** Nombre del parámetro.
  - **Nombre de la métrica:** Nombre de la métrica que define los posibles valores que puede tomar un parámetro.
  - **Tipo de métrica:** La métrica puede ser de distintos tipos: simple (entera, real, cadena de caracteres, enumerada, etc.) o estructurada (conjunto de elementos con una métrica simple, tupla de distintos valores con métrica simple, etc.)
- **Extensión(es).** Al igual que la parte fija del patrón, describen el objetivo y la propiedad a cumplir, y en caso necesario también pueden contener parámetros, que podrán tomar valores específicos para un cierto proyecto de selección, dependiendo de la métrica que tengan asignada (principio 44 [262]).

- **Comentarios:** Información extra sobre el requisito funcional que no puede ser descrita en los campos anteriores y se quiere dejar registrada.

En la figura 3-B se muestra la plantilla que se sigue para la estructuración de un patrón de requisitos funcionales de ERP.

<b>Identificador:</b> <número de patrón> <b>Nombre:</b> <nombre del patrón>			
<b>Módulo</b>	< módulo ERP >		
<b>Palabras clave</b>	<lista de palabras clave>		
<b>Versión</b>	<número de versión> (<fecha>)		
<b>Autor</b>	<autor de la versión> (<organización>)		
<b>Fuente</b>	<fuente de la versión actual>		
<b>Objetivo del Patrón</b>	<propósito general del patrón>		
<b>Descripción</b>	<descripción del patrón>		
<b>Parte fija del patrón</b>	<i>Objetivo del patrón &lt;debe&gt; propiedad a cumplir {variables}</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	{parámetro}	{métrica}	{tipo de métrica}
<b>Extensión</b>	<b>Identificador:</b> <número de la extensión del patrón> <b>Nombre:</b> <nombre de la extensión del patrón>		
	<i>Objetivo de la extensión1 &lt;debe&gt; propiedad a cumplir {variables}</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	{parámetro}	{métrica}	{tipo de métrica}
	...		
<b>Comentarios</b>	<comentarios>		

Figura 3-B: Plantilla de patrones funcionales para ERP

### 3.1.1 Clasificación de los patrones

Con el fin de clasificar los patrones, y así disponer de una estructuración del catálogo, nos proponemos utilizar las características de calidad del modelo ISO / IEC 9126-1 [143] que hemos descrito en el estado del arte. Ésta clasificación puede facilitar búsqueda e identificación y estructuración de los patrones durante el proceso de definición de requisitos dentro de un proyecto concreto.

Las características de calidad en esta extensión del modelo (características, atributos y sub-características) son de carácter general para cualquier aplicación de software empresarial.

En el ISO/IEC 9126-1 los factores de calidad funcionales deberían clasificarse bajo la característica *functionality*, y subcaracterística *suitability*. Que corresponden a la adecuación del producto software para proveer una funcionalidades determinadas (ver figura 3-C).

Characteristics	Subcharacteristics
Functionality	suitability
	accuracy
	interoperability
	security
	functionality compliance

Figura 3-C: Clasificación de functionality en el ISO/IEC 9126-1.

Por otro lado, es importante señalar que el patrón se pueden clasificar (de ser necesario) en más de un lugar en el modelo de calidad.

En la figura 3-D se muestra el modelo de calidad a seguir propuesto para la clasificación de los patrones de requisitos funcionales de ERP.

		Sub-características			Definiciones
Características	Funcionalidad	Nivel 1	Nivel 2	Nivel 3	
		Adecuación	Adecuación básica	Ventas	Requisitos funcionales propios del ERP
				Mantenimiento	
				Proyectos	
				Compras	
				Logística	
				Capital humano	
				Finanzas	
				Producción	
				Aplicaciones cruzadas	
			Adecuación específica de la organización		Requisitos funcionales propios de la organización que selecciona el ERP

Figura 3-D: Modelo de calidad para clasificar patrones de requisitos funcionales para ERP

## 3.2 Consideraciones previas

Para llegar a obtener la primera heurística del método propuesto en esta tesis de máster hemos seguido un proceso *botton-up*. Es decir, nuestra idea ha sido partir de los requisitos para construir los patrones. Existen propuestas como la descrita en [264] para realizarlo siguiendo un proceso *top-down* (de lo general a lo particular), por lo que no descartamos el hecho de realizar un análisis posterior de este tipo que por analogía podría ser aplicado de forma similar y enriquecer el método propuesto. Actualmente se han realizado unas primeras pláticas para una posible colaboración con el profesor Geert Poels de la universidad de Ghent con el fin de estudiar esta línea de trabajo.

La utilización de un enfoque *bottom-up* se ha podido realizar gracias a la colaboración entre CITI (Centre d'Innovation par les Technologies de l'Information) y el grupo GESSI. El centro CITI interviene en distintos proyectos de selección de OTS, que en muchos casos son ERP, por lo que dispone de distintos libros de requisitos del dominio de los ERP. Así, los patrones propuestos tendrán la ventaja de que se definirán a partir de requisitos reales.

### 3.2.1 CITI y su papel en la selección de ERP

CITI fue creado en 1987, y es el departamento de I+D del Centro de Investigación Pública Henri Tudor de Luxemburgo. A continuación se describe el trabajo de CITI y cuál es su papel en los proyectos de selección de OTS en qué interviene. Después expondremos los objetivos de la colaboración actual existente entre CITI y el grupo GESSI.

La finalidad de CITI es favorecer la innovación tecnológica de las empresas situadas en Luxemburgo, brindando servicios de consultoría y transferencia de conocimiento en los sectores privados y públicos. Actualmente, dispone de cien ingenieros dedicados a la investigación y desarrollo, quienes se dedican principalmente a tratar los siguientes temas:

- Seguridad en sistemas de información
- E-learning, gestión del conocimiento e investigación cooperativa
- Estándares de interoperabilidad y e-business
- Calidad y certificación de servicios informáticos
- Estudios estadísticos y prospectivos de la economía del conocimiento
- Software libre

CITI realiza consultoría durante las adquisiciones de componentes software, tanto de empresas pequeñas y medianas, así como del gobierno; en este trabajo colaboran con la red CASSIS, certificándola como capacitada para la correcta realización el proceso de adquisición de paquetes de software.

Por su parte la red CASSIS, es una red que agrupa consultores en TI, siempre coordinados por CITI. Así, CITI define el proceso y método a seguir durante la adquisición de componentes de software y los consultores de CASSIS siguen dicho proceso.

Para seguir este proceso de trabajo cuentan con una herramienta de software llamada OPAL [265] desarrollada por ellos mismos, la cual les ayuda entre otras tareas a:

- Identificar objetivos y requisitos.
- Generar los libros de requisitos.
- Preparar la construcción del cuestionario para la generación del call for tenders.
- Analizar y elegir las ofertas obtenidas del call for tenders

A continuación se describe el proceso de trabajo utilizado por CITI/CASSIS así como la utilización de OPAL (ver figura 3-E).

#### ELICITACIÓN DE REQUISITOS:

- Contactar con el cliente interesado en adquirir un componente o un sistema.
- Identificar el tipo de software que el cliente necesita.
- Ayudar al cliente en la determinación de los requisitos
- Introducir los requisitos definidos en la herramienta OPAL (desarrollada por CITI) formando el libro de requisitos del proyecto de adquisición. Los libros de requisitos contienen toda la información obtenida en la fase de elicitación de requisitos, diferenciándolos en funcionales, no funcionales, y de proyecto o también llamados no técnicos.



### GENERACIÓN DE CALL FOR TENDERS:

- Generar con ayuda de OPAL un call for tenders, con un cuestionario que más adelante será enviado a los proveedores potenciales que ofrecen productos que podrían adecuarse a las necesidades del cliente.

### EXPLORACIÓN DEL MERCADO:

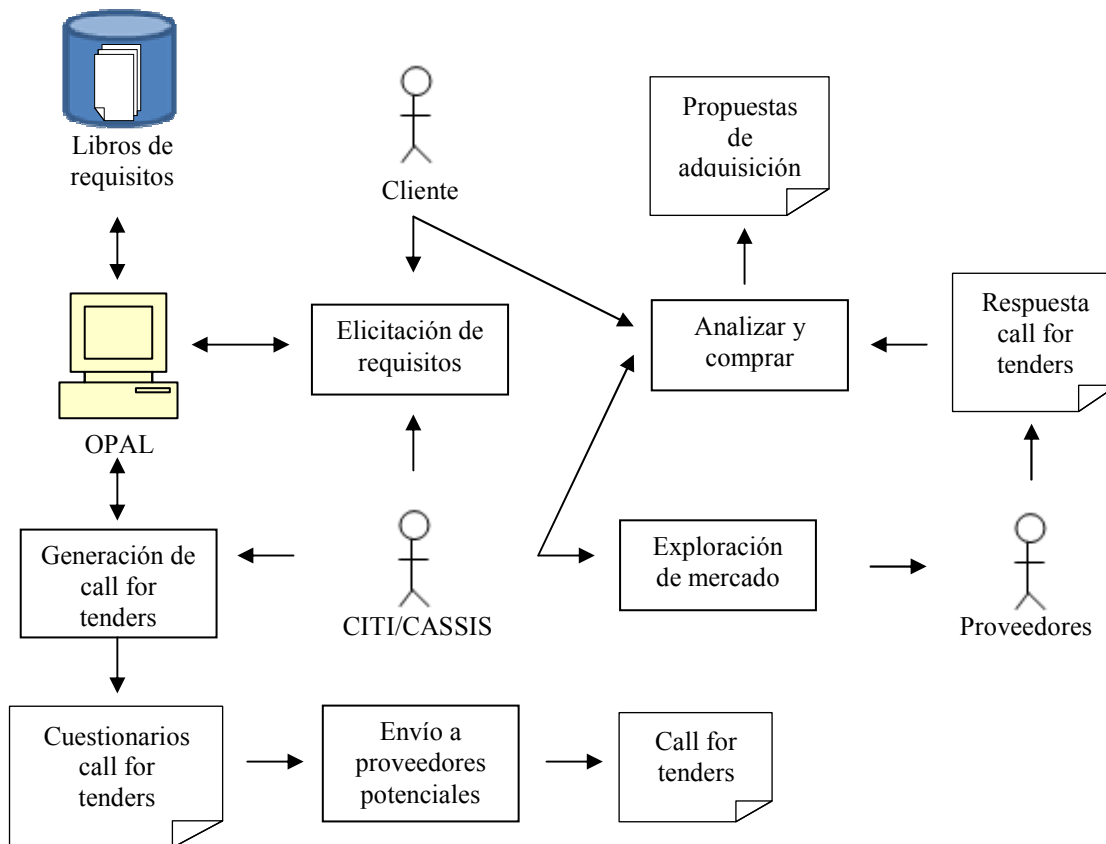
- Explorar el mercado, si es necesario, en busca de proveedores potenciales.

### ENVÍO A PROVEEDORES POTENCIALES:

- Mandar el call for tenders a los proveedores potenciales que se considera que pueden estar interesados en el proyecto.

### ANÁLISIS Y COMPARACIÓN DE LAS RESPUESTAS RECIBIDAS:

- Analizar y comparar las propuestas recibidas.
- Informar al cliente sobre las propuestas de adquisición que mejor se adecuan a sus necesidades.



**Figura 3-E: Forma de trabajo CITI/CASSIS**

### 3.2.2 Colaboración entre CITI y GESSI

Tal como hemos dicho, una de las finalidades de CITI es ofrecer asesoramiento a empresas y consultores para ayudar en la selección de componentes de software de unos mismos dominios. En sus años de experiencia, los miembros de CITI han observado que en muchas ocasiones los requisitos que se obtienen son iguales o muy parecidos principalmente en el caso de proyectos de selección en un mismo dominio de software. Hasta el momento no utilizan un proceso claro de reutilización, y se limitan simplemente a copiar y pegar los requisitos de proyectos anteriores para ser reescritos y adaptados a los nuevos proyectos.

Por su parte GESSI lleva a cabo la investigación en muchos campos de la ingeniería de software, dando especial énfasis en la adquisición de componentes OTS, la ingeniería de requisitos, la construcción de modelos de calidad para dominios de software. Por parte del grupo GESSI estamos involucrados Xavier Franch y Carme Quer como supervisores, y Oscar Medez y yo misma.

La colaboración de CITI con el grupo GESSI tiene como objetivo:

*La definición de un proceso de reutilización de requisitos que permita ahorrar tiempo en la generación de libros de requisitos y mejorar la calidad de los libros obtenidos*

Para conseguir dicho objetivo, se ha acordado la investigación de si se podría conseguir mediante la definición de patrones de requisitos, mediante los cuales se pueda crear una forma estándar en la creación y redacción de requisitos, mejorando así el proceso de reutilización, ahorrando tiempo y mejorando la calidad.

CITI puede aportar para realizar la investigación los libros de requisitos que CITI que ha obtenido durante su trabajo en procesos de selección de OTS durante los últimos 8 años.

En el trabajo realizado hasta ahora, en el marco de la colaboración, se han realizado los primeros pasos para establecer un proceso para la extracción de patrones de requisitos de dichos libros. Oscar Méndez miembro de GESSI se encuentra trabajando en la estructura que deben tener los patrones de requisitos en el caso concreto de requisitos no funcionales y no técnicos [263], así como en la clasificación de dichos patrones para facilitar su identificación y reutilización.

Por mi parte, el enfoque de esta tesis de máster es en la construcción de un método para la obtención de patrones de requisitos funcionales de ERP y la construcción de unas primeras heurísticas de un primer catálogo de patrones de requisitos funcionales para el dominio de los ERP a partir de los libros de requisitos disponibles y su validación en proyectos en los que CITI intervenga. De dicha validación se podrá obtener datos que servirán para hacer evolucionar el proceso de extracción de patrones, de forma que tenga en cuenta también el feed-back de las personas que usen los patrones, los nuevos requisitos identificados que no provengan de la aplicación de patrones y las estadísticas de uso de los patrones.

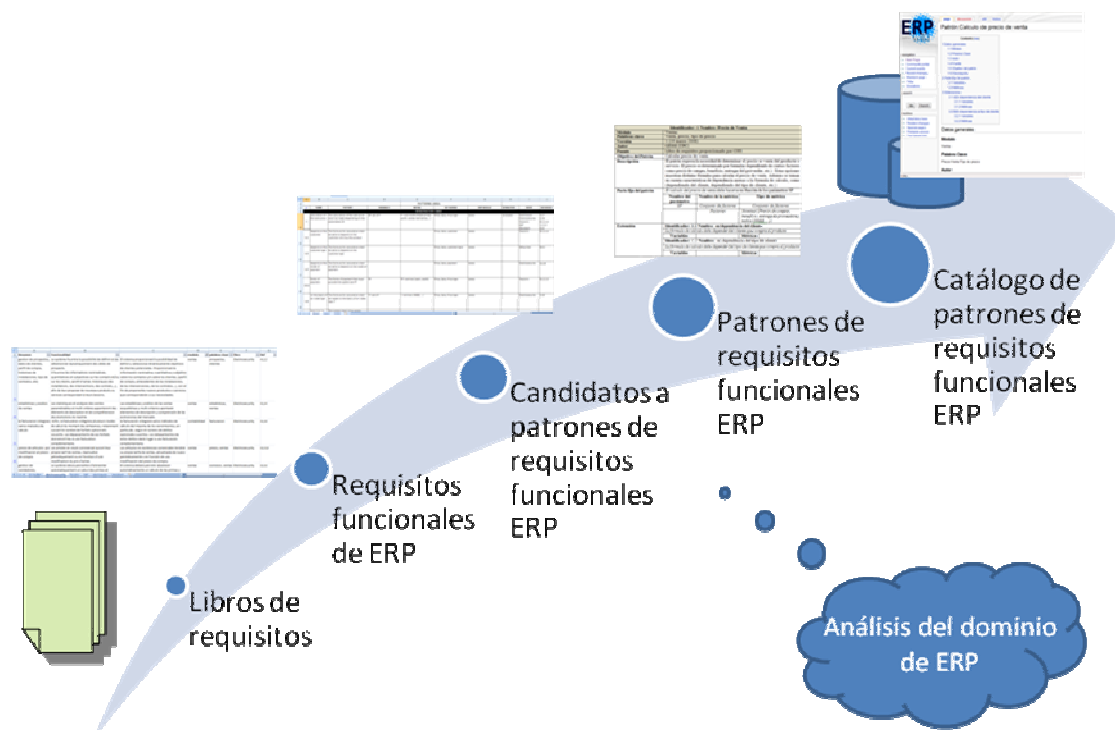
### 3.3 Método

En esta sección se exponen los pasos que se han seguido para la creación del primer catálogo de requisitos para ERP presentado en esta tesis de máster.

Primeramente, antes de empezar el proceso de extracción de patrones, y teniendo en cuenta que la extracción debía ser de patrones funcionales del dominio de los ERP, se procedió al estudio del dominio. Describiremos en la sección 3.3.1 como se realizó el análisis del dominio de los ERP, lo que nos permitió conocer todas las características y funcionalidades relativas a dicho dominio descrito.

Una vez tuvimos suficiente conocimiento del dominio, se realizó el proceso de extracción de patrones, descrito en la sección 3.3.2. En esta sección se muestran los pasos del método propuesto para la realización de dicho catálogo.

Sin embargo, el análisis del dominio se puede realizar antes o durante el análisis de los requisitos funcionales. En este caso lo hemos hecho antes de realizar el análisis de los requisitos funcionales para poder establecer las dependencias entre módulos ERP y sus correspondientes funcionalidades.



**Figura 3-F: Método propuesto.**

En la figura 3-F se muestran los pasos que se han seguido para la realización de unas primeras heurísticas para un primer catálogo de patrones de requisitos funcionales de ERP. Este catálogo consiste en un conjunto estructurado de los patrones de requisitos funcionales previamente obtenidos, siguiendo las fases descritas a continuación:

1. Realizar un *análisis de dominio de los ERP*.
2. *Extraer los requisitos funcionales* de los libros de requisitos.
3. Realizar un *análisis semántico y refinar* los requisitos funcionales.
4. Insertar en el *catálogo de candidatos a patrones* aquellos requisitos aptos como candidatos a convertirse en patrones. Ya sea como candidato nuevo o como candidato asociado según el resultado de la búsqueda de coincidencias en el catálogo de candidatos.
5. Crear y/o refinar el patrón e insertarlo o actualizarlo en el *catálogo de patrones*.

A continuación describimos los pasos que se han seguido para la construcción de las primeras heurísticas de nuestro primer catálogo de patrones de requisitos funcionales para ERP.

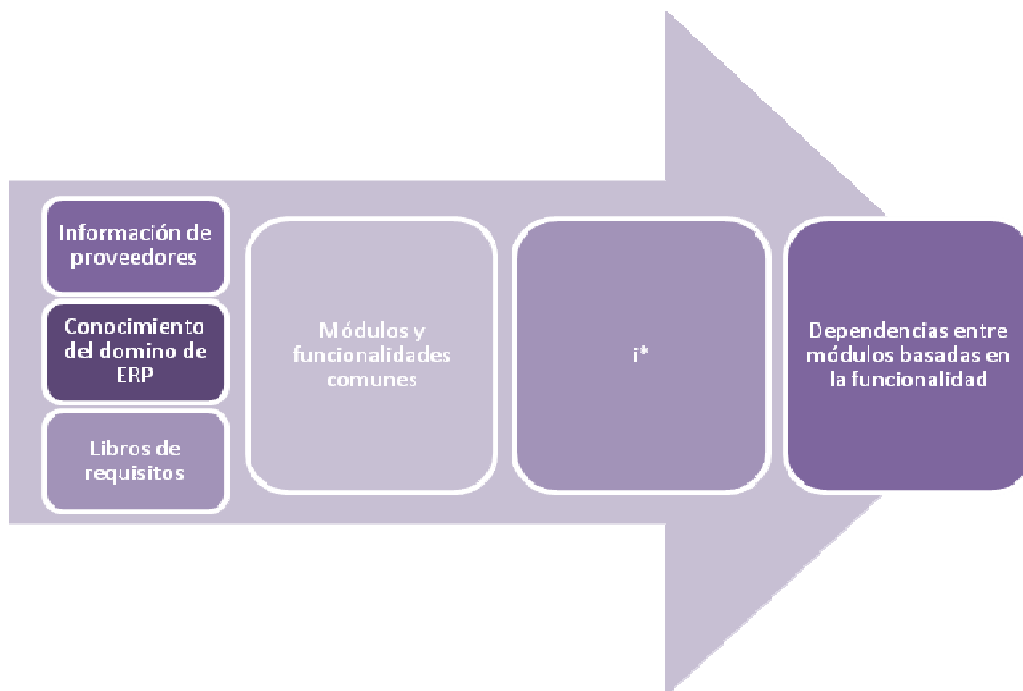
### **3.3.1 Análisis del dominio de ERP**

En el estado del arte podemos observar que la modularidad de un ERP depende principalmente en cómo se compra y se implementa el ERP, ya que puede ser que se requieran todos los módulos a la vez o solamente algunos de ellos. Los módulos pueden trabajar independientemente o en conjunto de forma integrada. Sin embargo existen algunas dependencias entre los módulos.

Con el fin de obtener estas dependencias, se ha llevado a cabo la identificación de los módulos comunes y las funcionalidades. Estas funcionalidades marcan dependencias claras permitiendo la comunicación y la dependencia entre módulos tal como se describe en el estado del arte. Una vez analizado el dominio de los ERP se realiza un modelo *i\** para representar estas dependencias.

En la figura 3-G se muestra el proceso que se ha seguido para la obtención de dependencias entre los módulos comunes en los distintos ERP. Para lo cual se realiza un estudio del dominio de los ERP analizando la información de dicho dominio, tal como se describe en el estado del arte.

Concretamente se ha analizado la información proporcionada por los proveedores de ERP, el conocimiento general del dominio de los ERP y la obtenida de los libros de requisitos proporcionados por CITI. Con toda esta información se han detectado módulos y funcionalidades comunes que nos permite representar las dependencias entre los módulos mediante modelos de actores y objetivos utilizando el lenguaje *i\**, y siguiendo la metodología RiSD [213].



**Figura 3-G: Obtención de dependencias entre módulos ERP**

Un modelo SD de  $i^*$  tal como se describe en el estado del arte consta de un conjunto de nodos que representan a actores y un conjunto de dependencias que representan las relaciones entre ellos, expresando que un actor (depende) depende de algunos otros (dependee) con el fin de obtener algún objetivo (dependum).

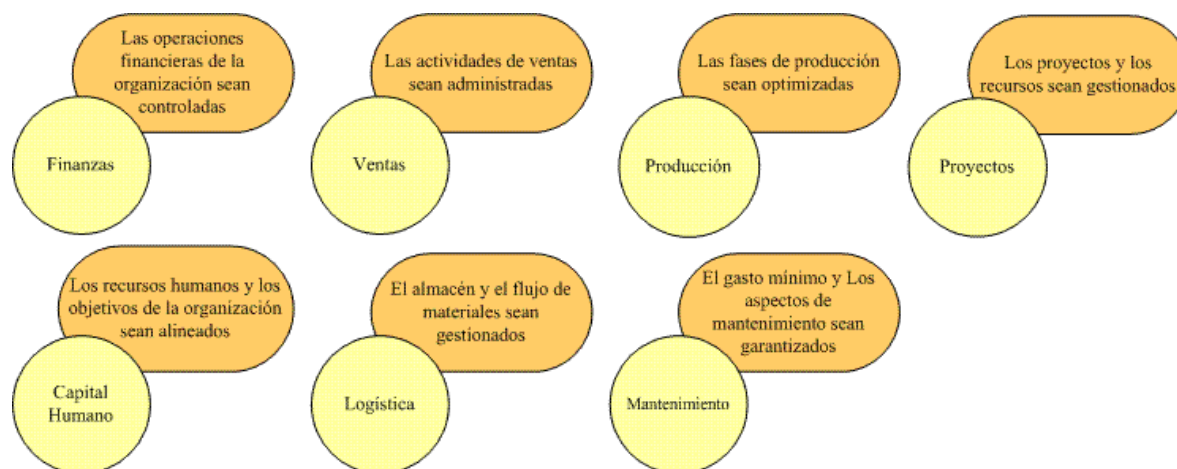
En base a la metodología RiSD [213], con el fin de utilizar una terminología estándar para realizar esta representación, la definición de los módulos que se ha utilizado en el modelo  $i^*$  es la descrita en el estado del arte.

En este caso realizaremos únicamente la construcción del sistema social (SD) centrándonos en la fase I de la metodología RiSD, para lo cual propone seguir las siguientes actividades:

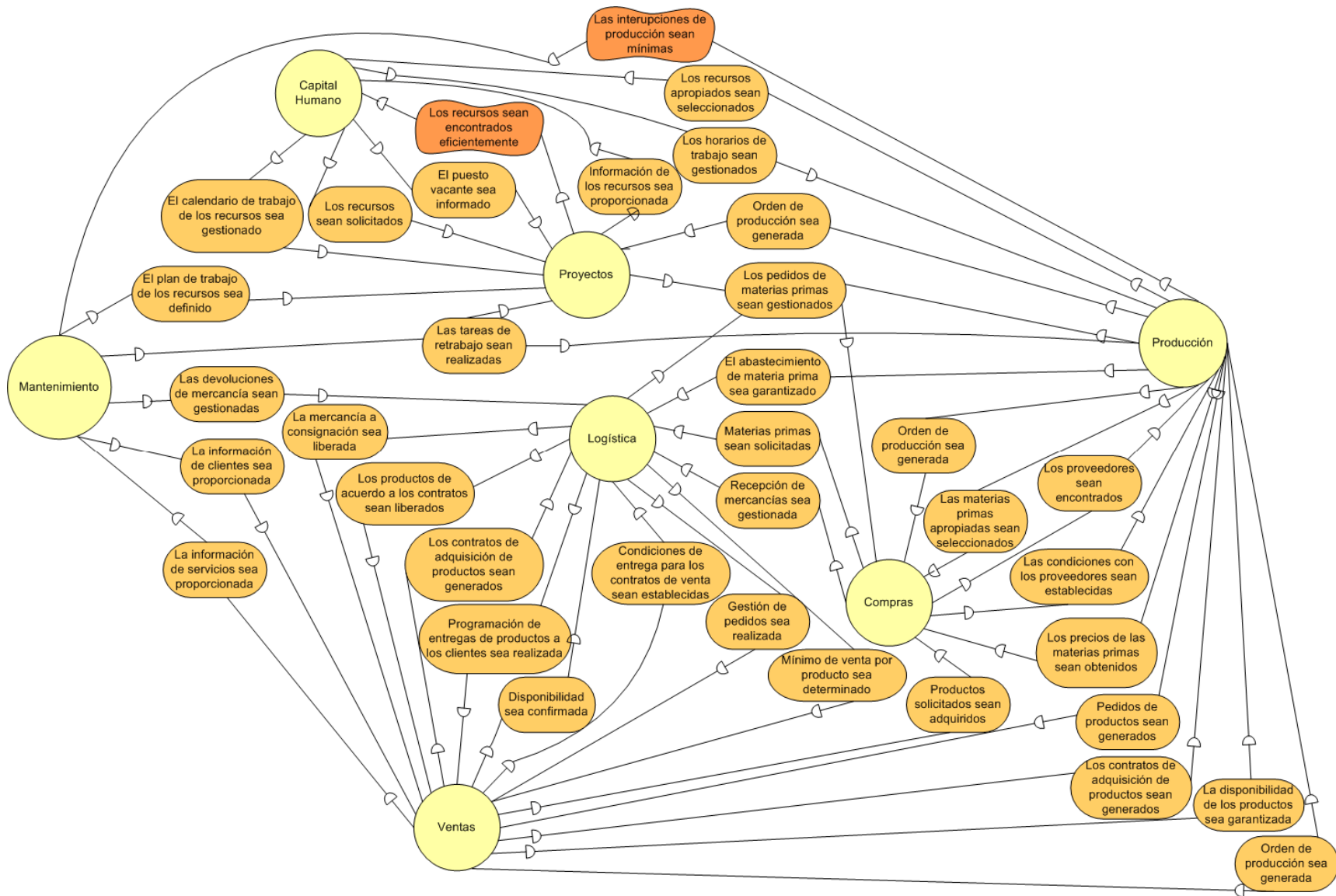
1. Identificar actores de salida. Consiste en identificar los actores principales y sus objetivos
2. Establecer dependencias objetivo entre actores. Las dependencias entre los actores se clasifican en dependencias tipo objetivo (goal) que posteriormente se confirman o se cambian de clasificación.
3. Clasificar dependencias añadidas. Precisando las dependencias obtenidas en la actividad anterior, se clasifican definitivamente los goal en: task, resource, goal o softgoal.
4. Analizar las consecuencias de las dependencias. Se comprueba si alguna de las dependee es capaz de satisfacer por sí solo las necesidades del dependum o si necesita la ayuda de otro agente, que puede existir ya, o todavía no (en este último caso, su objetivo debe ser declarado en primer lugar, como se hizo en actividad 1).
5. Refinar el sistema social. Se realiza iterando entre las actividades 3 y 4 según sea necesario. Este proceso es ambiguo, sin embargo podemos parar una vez que se en la última iteración se han identificado únicamente tasks y resources.

De estas actividades y para el fin de esta tesis de máster, hemos realizado las actividades 1 y 2. Dejando la precisión y el refinamiento como pasos futuros.

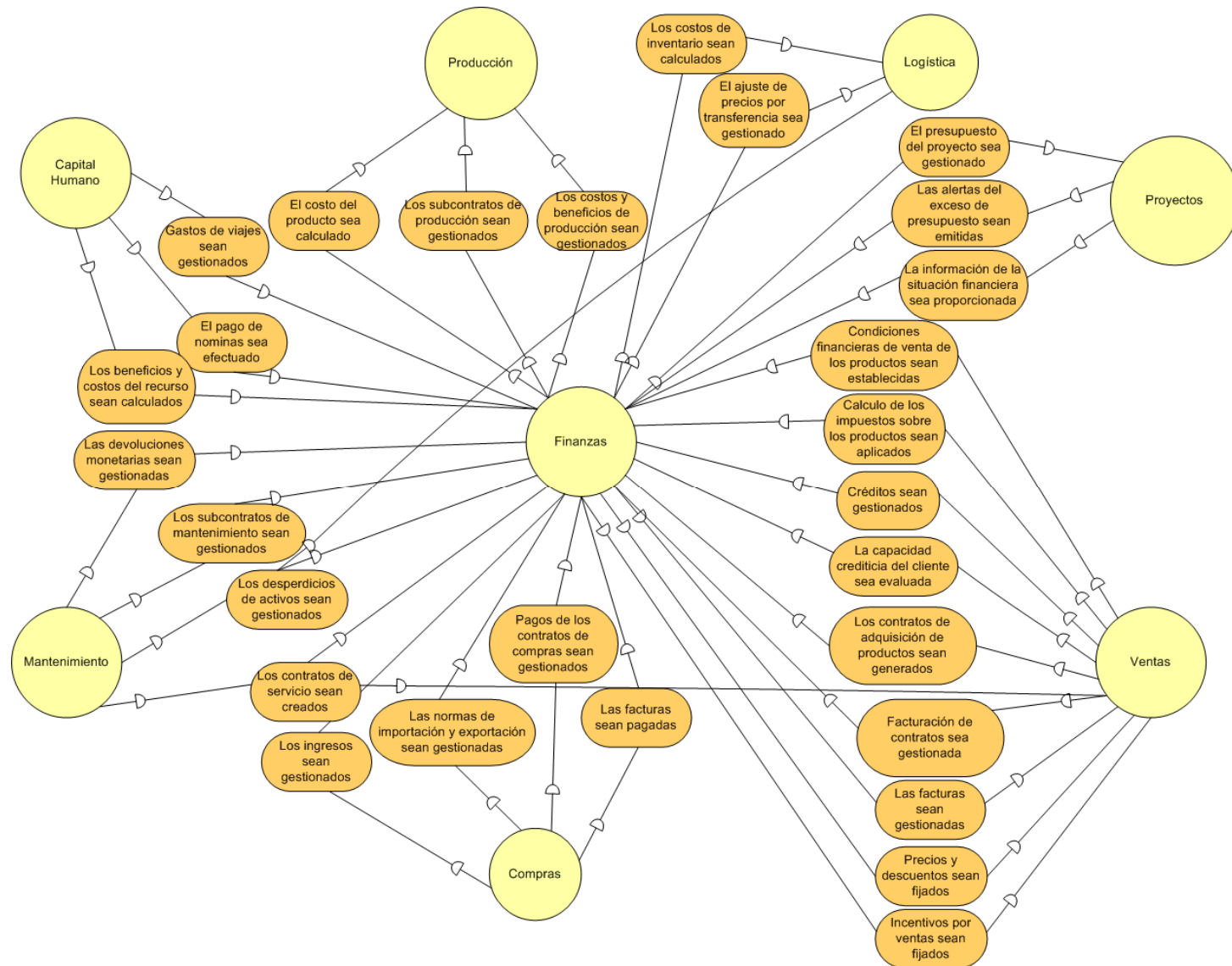
**Actividad 1:** Hemos identificado los principales actores y sus objetivos, en este caso de cada modulo básico de los ERP descrito en el estado del arte (ver figura 3-H).



**Figura 3-H: Actores principales y objetivos.**



**Figura 3-I: Modelo SD  $i^*$ . Relación entre módulos básicos de un ERP (excepto Finanzas)**



**Figura 3-J: Modelo SD i\*. Relación entre módulos básicos de un ERP (Finanzas)**



**Actividad 2:** Al establecer las dependencias entre dichos actores el punto crucial es identificar aquellas que son realmente importantes. Esto es inevitablemente subjetivo, por lo que en nuestro caso hemos tratado de responder a la pregunta ¿Qué necesita un módulo ERP de los demás módulos del sistema para su correcto funcionamiento?

Por motivos prácticos hemos separado la relación entre finanzas y los demás módulos en dos figuras. En la figura 3-I se muestra el análisis de las dependencias entre los módulos básicos de los ERP (a excepción de finanzas). En la figura 3-J se muestra el análisis de las dependencias entre finanzas y los demás módulos básicos de los ERP. Ambas, se han expresado en el lenguaje *i\**. Teniendo en cuenta que:

- El modulo de Aplicaciones cruzadas, es un modulo que no aporta funcionalidades propias de negocio, sino funcionalidades de configuración de la aplicación. Por otra parte, el modulo de Análisis, es un modulo que no aporta lógica de negocio ya que es un sistema de reportes y consultas de las acciones realizadas en el resto de módulos. Por consiguiente no son considerados en el diagrama SD de *i\**.
- No hay un consenso claro en los distintos proveedores de ERP acerca de si ubicar cuentas por pagar y cuentas por cobrar como parte de los módulos de compras y ventas respectivamente, como parte del módulo de finanzas, o como módulos independientes. Esto se debe a que son acciones de comunicación entre módulos. Podríamos decir que son dependencias funcionales implementadas como una nueva funcionalidad. En este caso están consideradas dentro del modulo de Finanzas y su dependencia esta expresada en el diagrama *i\**.

De lo que podemos observar que en este primer modelo se distinguen dependencias entre actores en forma de objetivo.

### 3.3.2 Método aplicado

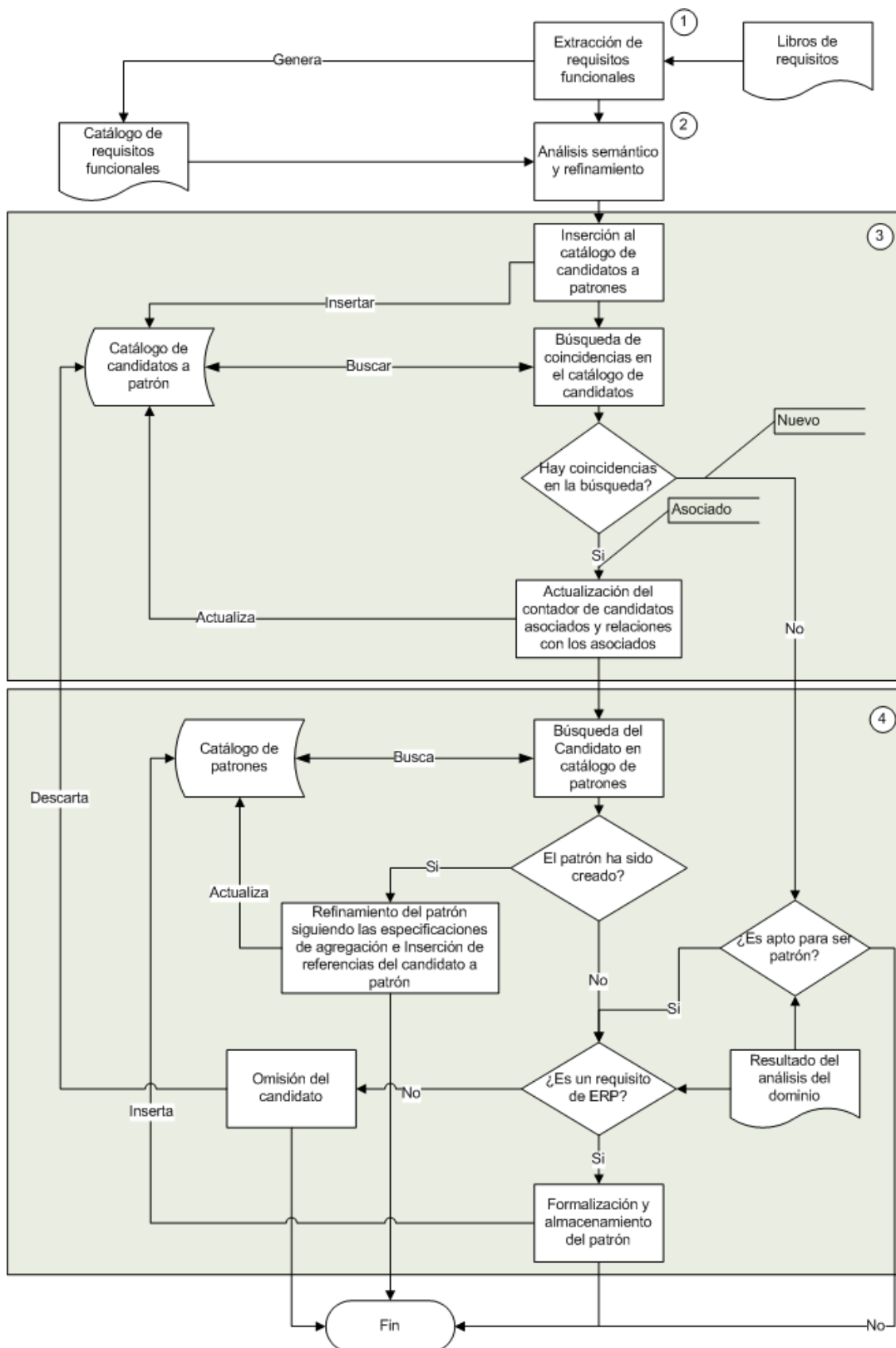


Figura 3-K: Pasos para la construcción de un catálogo de patrones

Las tareas de que consta el método son las siguientes:

- Tarea 1. Extracción de requisitos funcionales.
- Tarea 2. Análisis semántico y refinamiento.
- Tarea 3. Inserción en el catálogo de candidatos a patrón.
  - Tarea 3.1 Búsqueda de coincidencias en el catálogo de candidatos.
  - Tarea 3.2 Validación de coincidencias y actualización o inserción en el catálogo de candidatos.
- Tarea 4. Inserción en el catálogo de patrones.
  - Tarea 4.1 Refinamientos del patrón.
  - Tarea 4.2 Formalización y almacenamiento en el catálogo de patrones.

En la figura 3-K puede verse el orden en que se desarrollan estas tareas, así como los flujos de información que circulan entre ellas. A continuación se puede encontrar una descripción de cada una de ellas.

### Tarea 1. Extracción de requisitos funcionales

Los requisitos plasmados en los libros son de tres tipos (funcionales, no funcionales y no técnicos) por lo que es necesario extraer únicamente aquellos que son funcionales, creando así un catálogo de requisitos funcionales. En este catálogo se transcriben los requisitos funcionales tal y como se encuentran en el libro de requisitos, así como su ubicación en dicho libro para, de esta forma, tener una referencia del origen del requisito y poder realizar futuras consultas (ver figura 3-L).

Funcionalidad	libro	Ref
Le système fournira la possibilité de définir et de sélectionner dynamiquement des cibles de prospects. Il fournira les informations nominatives, quantitatives et subjectives sur les contacts et/ou sur les clients, (profil d'achat, historiques des installations, des interventions, des contrats,...), afin de leur proposer de nouveaux produits ou services correspondant à leurs besoins.	Electrosecurity	4.2.2
Les statistiques et analyses des ventes paramétrables et multi critères apporteront des éléments de description et de compréhension des évolutions du marché.	Electrosecurity	4.2.4

Figura 3-L: Extracción de requisitos funcionales

### Tarea 2. Análisis semántico y refinamiento

Los requisitos funcionales se expresan en lenguaje natural (ver figura 3-M).

A intervalos regulares, el precio de coste de los productos se vuelve a calcular. Por lo que tratando de establecer un costo lo más amplia posible, mezclando además de las materias primas y mano de obra, otros conceptos como:

- Suministros (cápsulas, etiquetas, botellas, corchos,...)
- La depreciación de la maquinaria,
- Energía,
- Los alquileres,
- Los costos del transporte entre los sitios
- Los servicios exteriores (envasado, transporte,...)
- Los costes de re envasado
- Otros gastos diversos de producción

**Figura 3-M: Requisito funcional**

Un mismo requisito puede estar expresado de forma distinta en cada uno de los libros o contener más de un requisito expresado en él. Por lo que es necesario estudiar cada requisito contenido en el catálogo de requisitos funcionales mediante un análisis semántico. Este análisis semántico (de momento manual) permite entender e identificar los conceptos expuestos en cada requisito y, separar dicho requisito en los diferentes requisitos encontrados en el (ver figura 3-N).

- El precio de coste de los productos se vuelve a calcular a intervalos regulares
- El precio de coste de los productos se establece lo más amplio posible, mezclando además de las materias primas y mano de obra, otros conceptos como:
  - Suministros (cápsulas, etiquetas, botellas, corchos ,...)
  - La depreciación de la maquinaria,
  - Energía,
  - Los alquileres,
  - Los costos del transporte entre los sitios
  - Los servicios exteriores (envasado, transporte ,...)
  - Los costes de re-emplazado
  - Otros gastos diversos de producción

**Figura 3-N: Separación de requisito funcional ERP**

Posteriormente se refina cada requisito del catálogo, para así obtener un resumen de requisitos funcionales que aclare (ver figura 3-O):

- La separación del requisito funcional.
- El objetivo del requisito funcional.
- El módulo ERP al que pertenece (basándonos en los módulos obtenidos en la sección 2.6.3).
- La asignación de palabras clave.

Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
Cálculo de precio de coste (frecuencia)	El precio de coste de los productos se vuelve a calcular a intervalos regulares	producción	precio coste cálculo frecuencia	StFeuillen	B.2.6
Cálculo de precio de coste (definición de fórmula)	El precio de coste de los productos se establece lo más amplio posible, mezclando además de las materias primas y mano de obra, otros conceptos como: - Suministros (cápsulas, etiquetas, botellas, corchos ,...) - La depreciación de la maquinaria, - Energía, - Los alquileres, - Los costos del transporte entre los sitios - Los servicios exteriores (envasado, transporte ,...) - Los costes de re-embalado - Otros gastos diversos de producción	producción	precio coste cálculo definición fórmula	StFeuillen	B.2.6

**Figura 3-O: Refinamiento de catálogo de requisitos funcionales**

### Tarea 3. Inserción en el catálogo de candidatos a patrón

Cada requisito refinado se va insertando en un catálogo de candidatos a patrones como un requisito candidato. Para ello hay que identificar si el requisito candidato ya existe en dicho catálogo buscando coincidencias (es decir, se identifican requisitos similares). Basándonos en las palabras clave para facilitar su búsqueda.

#### Tarea 3.1 Búsqueda de coincidencias en el catálogo de candidatos

Esta búsqueda de coincidencias se hace con la finalidad de encontrar requisitos aptos a formalizarse como patrones (ver figura 3-P). Para agregar el requisito como candidato en el catálogo tenemos que validar si se han encontrado coincidencias.



**Figura 3-P: Identificación de posibles patrones**

#### Tarea 3.2 Validación de coincidencias y actualización o inserción en el catálogo de candidatos

- En caso de *No encontrar* coincidencias. El requisito es agregado al catálogo de candidatos como un *candidato nuevo* (ver figura 3-Q). Posteriormente se analiza (en base al resultado del análisis de dominio) si el candidato a pesar de ser un candidato nuevo, es apto para ser patrón.

- En caso de *encontrar* coincidencias. El requisito, éste es agregado como *candidato asociado*. Esta asociación se refleja actualizando el contador de candidatos asociados y las relaciones con los asociados.

id	#	rel	Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
1	0	0	Cálculo de precio de coste (definición de fórmula)	El precio de coste de los productos se establece lo más amplio posible, mezclando además de las materias primas y mano de obra, otros conceptos como: - Suministros (cápsulas, etiquetas, botellas, corchos ,...) - La depreciación de la maquinaria, - Energía, - Los alquileres, - Los costos del transporte entre los sitios - Los servicios exteriores (envasado, transporte ,...) - Los costes de re-ensado - Otros gastos diversos de producción	producción	precio coste cálculo definición fórmula	StFeuillen	B.2.6

**Figura 3-Q: Catálogo de candidatos a patrones funcionales (nuevo candidato)**

### Tarea 3.3 Actualización del contador de candidatos asociados y relaciones con los asociados

Esta actualización se refleja escribiendo el identificador del candidato con el que ha coincidido (ver figura 3-R) y aumentando el número de coincidencias de este candidato, actualizando el catálogo de candidatos a patrón. Con estas coincidencias se logra obtener la parte fija y las extensiones del patrón mediante un proceso iterativo. Posteriormente se debe analizar si para el candidato con el que se ha coincidido existe un patrón creado.

id	#	rel	Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
1	0	0	Cálculo de precio de coste (definición de fórmula)	El precio de coste de los productos se establece lo más amplio posible, mezclando además de las materias primas y mano de obra, otros conceptos como: - Suministros (cápsulas, etiquetas, botellas, corchos ,...) - La depreciación de la maquinaria, - Energía, - Los alquileres, - Los costos del transporte entre los sitios - Los servicios exteriores (envasado, transporte ,...) - Los costes de re-ensado - Otros gastos diversos de producción	producción	precio coste cálculo definición fórmula	StFeuillen	B.2.6
1	1	1	Cálculo de precio de coste (frecuencia)	El precio de coste de los productos se vuelve a calcular a intervalos regulares	producción	precio coste cálculo frecuencia	StFeuillen	B.2.6

**Figura 3-R: Catálogo de candidatos a patrones funcionales (candidato asociado)**

#### **Tarea 4. Inserción en el catálogo de patrones.**

La inserción en el catálogo de patrones depende del tipo de candidato que se está analizando, es decir, si se trata de un candidato nuevo o asociado.

##### **Tarea 4.1 Candidato nuevo**

Una vez que se ha insertado un requisito como candidato nuevo en el catálogo de candidatos, debemos validarlo en base al resultado del análisis de dominio con el fin de conocer si es apto a ser patrón. Ya que pudiera darse el caso de que a pesar que no existe un requisito igual en el catálogo de candidatos, se trate de un candidato propio del dominio de la organización existente en un módulo de ERP, no necesariamente considerado como básico.

##### **Tarea 4.1.1 Validación de candidatos aptos a patrón**

Tomando como base el resultado del análisis de dominio se analiza el candidato a ser patrón validando su importancia en el dominio. Del resultado de esta validación tenemos que:

- En caso de *no ser* un candidato apto a ser patrón se finaliza el proceso.
- En caso *ser* un candidato apto a ser patrón se valida si se trata de un requisito funcional de ERP. (siguiendo la validación de requisitos de ERP descrita en la tarea 4.1.2)

##### **Tarea 4.1.2 Validación de requisitos funcionales ERP**

Se trata de analizar si un candidato que es apto a ser patrón pertenece a las funcionalidades propias de un ERP según los resultados obtenidos del análisis del dominio. Por lo que:

- En caso de *no ser* requisito funcional de ERP se descarta el candidato y termina el proceso.
- En caso de *ser* un requisito funcional de ERP se formaliza el requisito siguiendo las especificaciones dándoles una estructura en forma de patrones (siguiendo la estructura y la plantilla de patrones descrita en 3.1). En la figura 3-Z se muestra el ejemplo de un patrón.

##### **Tarea 4.1.3 Formalización y almacenamiento en el catálogo de patrones.**

Una vez que se ha creado el patrón siguiendo la estructura y la plantilla de patrones descrita en la sección 3.1. Finalmente se agrega al repositorio del catálogo de patrones de requisitos funcionales de ERP (ver figura 3-S).



Figura 3-S: Repositorio de patrones de requisitos funcionales ERP.

## Tarea 4.2 Candidato asociado

Una vez que se ha insertado un requisito como candidato asociado en el catálogo de candidatos, debemos validar si existe un patrón creado para dicho candidato y sus asociaciones en el catálogo de patrones.

### Tarea 4.2.1 Búsqueda de candidatos en el catálogo de patrones

Esta búsqueda de coincidencias se hace con la finalidad de encontrar patrones creados con anterioridad que coincidan con lo descrito en el candidato asociado, por lo que es necesario validar la existencia de dicho patrón.

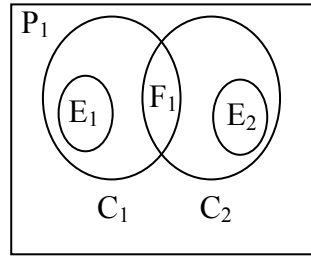
### Tarea 4.2.2 Validación de coincidencia de patrón

- En caso de *no existir* un patrón creado, se analiza (en base al resultado del análisis de dominio) si el candidato es un requisito funcional de ERP. (siguiendo la validación de requisitos de ERP descrita en la tarea 4.1.2)
- En caso de *existir* un patrón creado, se refina el patrón (siguiendo las *especificaciones de refinamiento del patrón* descritas en la tarea 4.2.3) y se actualiza el repositorio de patrones funcionales de ERP junto con las referencias del candidato.

### Tarea 4.2.3 Refinamiento del patrón

La diferencia entre agregar un candidato formalizado como extensión o como parte fija del patrón depende de la parte de los requisitos candidatos (C1, C2) en la que ambos coincidan, es decir:



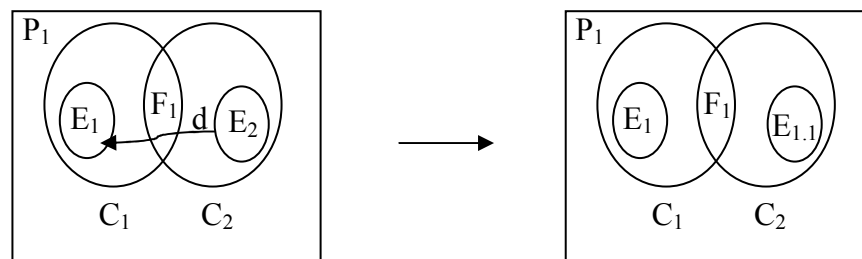


**Figura 3-T: Partes de un patrón de requisitos**

En base a dos candidatos  $C_1$  y  $C_2$  se construye por primera vez el patrón  $P_1$ , teniendo en cuenta la parte fija  $F_1$  y las extensiones  $E_1$  y  $E_2$  (en caso de que  $E$  exista), donde (ver figura 3-T):

- $E_1$  es la parte específica contenida únicamente en el candidato  $C_1$  que depende directamente de  $P_1$ .
- $E_2$  es la parte específica contenida únicamente en el candidato  $C_2$  que depende directamente de  $P_1$ .
- $F_1$  es la parte común entre  $C_1$  y  $C_2$ .
- $P_1$  es el conjunto de  $F$ ,  $E_1$  y  $E_2$ .

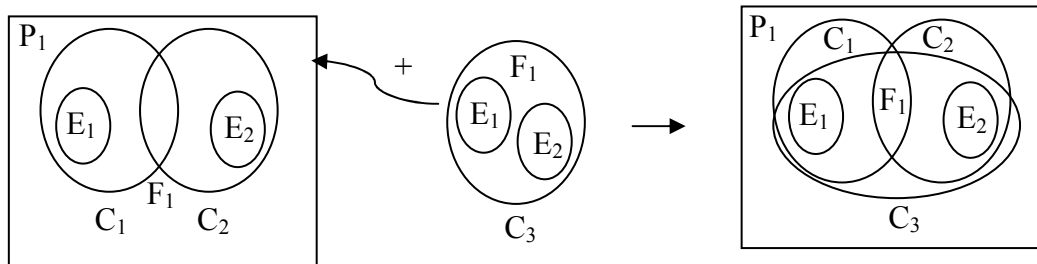
Tomando en cuenta que las extensiones se asocian según su dependencia (de quien dependen). El hijo siempre depende del padre. Por tanto si  $E_2$  depende directamente de  $E_1$ , la extensión de  $C_2$  sería  $E_{1.1}$  (ver figura 3-U).



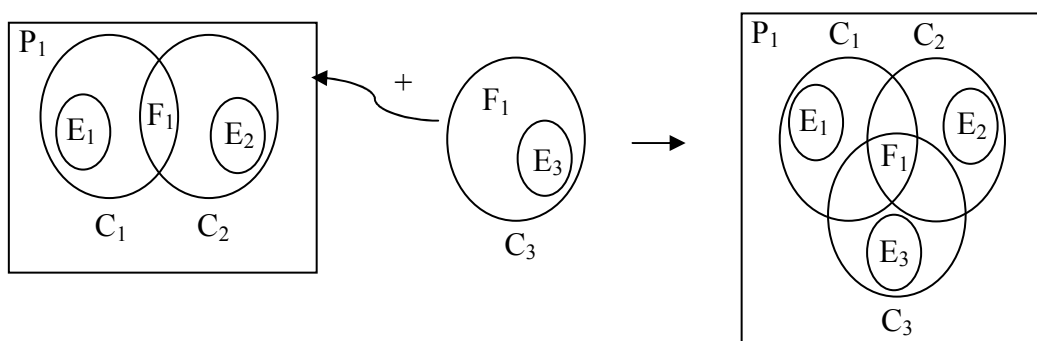
**Figura 3-U: Dependencia entre extensiones**

Cuando se requiere añadir un tercer candidato  $C_3$  hay que determinar si éste se agregará como parte de la parte fija y/o como una extensión del patrón, por lo que se debe tomar en cuenta:

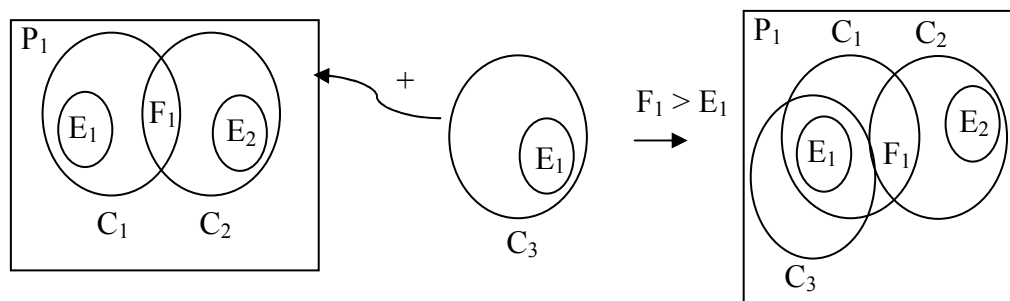
- **Caso1:**  $C_3$  está incluido por completo en  $P$ , por lo que únicamente se agrega la referencia del libro de requisitos donde se ha obtenido (ver figura 3-V).



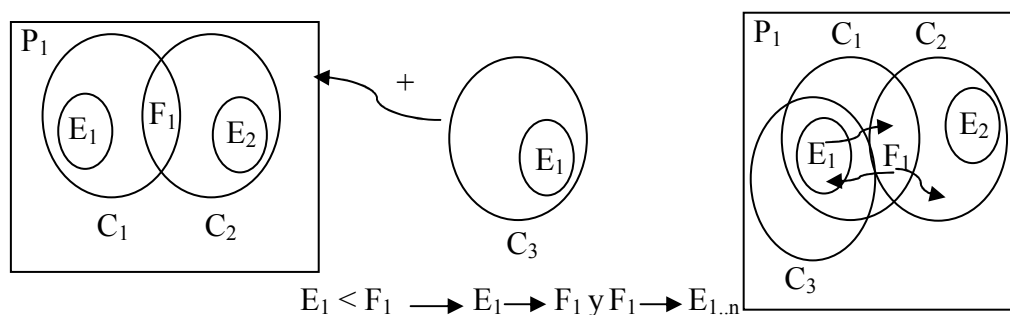
**Figura 3-V: Caso 1. Candidato totalmente contenido en el patrón.**



**Figura 3-W: Caso 2. Una parte específica no incluida en el patrón.**



**Figura 3-X: Caso 3. Una parte específica sin incluir  $F_1$  en el patrón ( $F_1 > E_1$ ).**



**Figura 3-Y: Caso 3. Una parte específica sin incluir  $F_1$  en el patrón ( $E_1 > F_1$ ).**

- **Caso2:**  $C_3$  contiene la parte fija del patrón  $F_1$  y una parte específica  $E_3$  que no está contenida en  $P_1$ . Por lo que  $E_3$  se agregaría a  $P$  tomando en cuenta la dependencia de las extensiones explicada anteriormente (ver figura 3-W).
- **Caso3:**  $C_3$  no contiene la parte fija del patrón  $F_1$ , pero si una parte específica  $E$  que está contenida en  $P_1$  como extensión ( $E_1$  o  $E_2$ ). Por lo que se debe tomar en cuenta la diferencia entre el número de referencias existentes en las coincidencias de  $F_1$  y de de la extensión con la que coincide, ya sea  $E_1$  o  $E_2$ .
  - Si el número de referencias en  $F_1$  es mayor se considerará dentro de la extensión del patrón con la cual coincide ya sea  $E_1$  o  $E_2$  (ver figura 3-X).
  - Si el número de referencias en  $E_1$  o  $E_2$  es mayor, se considerará ya sea  $E_1$  o  $E_2$  y parte con la cual coincide como la nueva parte fija del patrón  $F_1$ , mientras  $F_1$  se consideraría como una extensión (ver figura 3-Y).

Identificador: 1 Nombre: Precio de Venta			
Módulo	Ventas		
Palabras clave	Venta, precio, tipo de precio		
Versión	1 (19 marzo 2008)		
Autor	GESSI (UPC)		
Fuente	Libro de requisitos proporcionado por CITI		
Objetivo del Patrón	Calcular precio de venta		
Descripción	El patrón expresa la necesidad de determinar el precio se venta del producto o servicio. El precio es determinado por fórmulas dependiendo de ciertos factores como precio de compra, beneficio, entregas del proveedor, etc.). Estas opciones muestran distintas fórmulas para calcular el precio de venta. Además se toman en cuenta características de dependencia anexas a la fórmula de cálculo, como (dependiendo del cliente, dependiendo del tipo de cliente, etc.)		
Parte fija del patrón	<i>El cálculo del precio de venta debe hacerse en función de los parámetros SP</i>		
	Nombre del parámetro	Nombre de la métrica	Tipo de métrica
	<i>SP</i>	<i>Conjunto de factores</i>	<i>Conjunto de factores</i>
		<i>Factores</i>	<i>Nominal {Precio de compra, beneficio, entrega de proveedores, índice INSEE ,...}</i>
Extensión	<b>Identificador: 1.1 Nombre: en dependencia del cliente</b>		
	<i>La fórmula de cálculo debe depender del cliente que compra el producto</i>		
	Nombre del parámetro	Nombre de la métrica	Tipo de métrica
	<b>Identificador: 1.2 Nombre: en dependencia del tipo de cliente</b>		
	<i>La fórmula de cálculo debe depender del tipo de cliente que compra el producto</i>		
	Nombre del parámetro	Nombre de la métrica	Tipo de métrica

Figura 3-Z: Ejemplo de patrón de requisitos funcionales ERP

### 3.4 Diagrama UML para la creación de un patrón

En la figura 3-AA se muestra un diagrama UML para la creación de patrones funcionales de ERP, en donde podemos observar que:

- Los libros de requisitos de CITI son una agrupación de requisitos funcionales, no funcionales y no técnicos.
- De la extracción de requisitos funcionales de dichos libros, obtenemos un catálogo de requisitos funcionales. Por lo que podemos decir que existen dos tipos de libros de requisitos, por una parte los libros de requisitos de CITI y por otra el catálogo de requisitos funcionales.
- Este catalogo de requisitos funcionales corresponde a una agrupación de requisitos funcionales.
- Para cada requisito funcional se extraen varios candidatos a patrón.
- El catálogo de candidatos a patrón corresponde a una agrupación de requisitos funcionales candidatos a patrón.
- Un candidato a patrón pasa a ser patrón cuando el tipo de requisito corresponde a un requisito de ERP y el número de apariciones en el libro de requisitos es mayor a una.

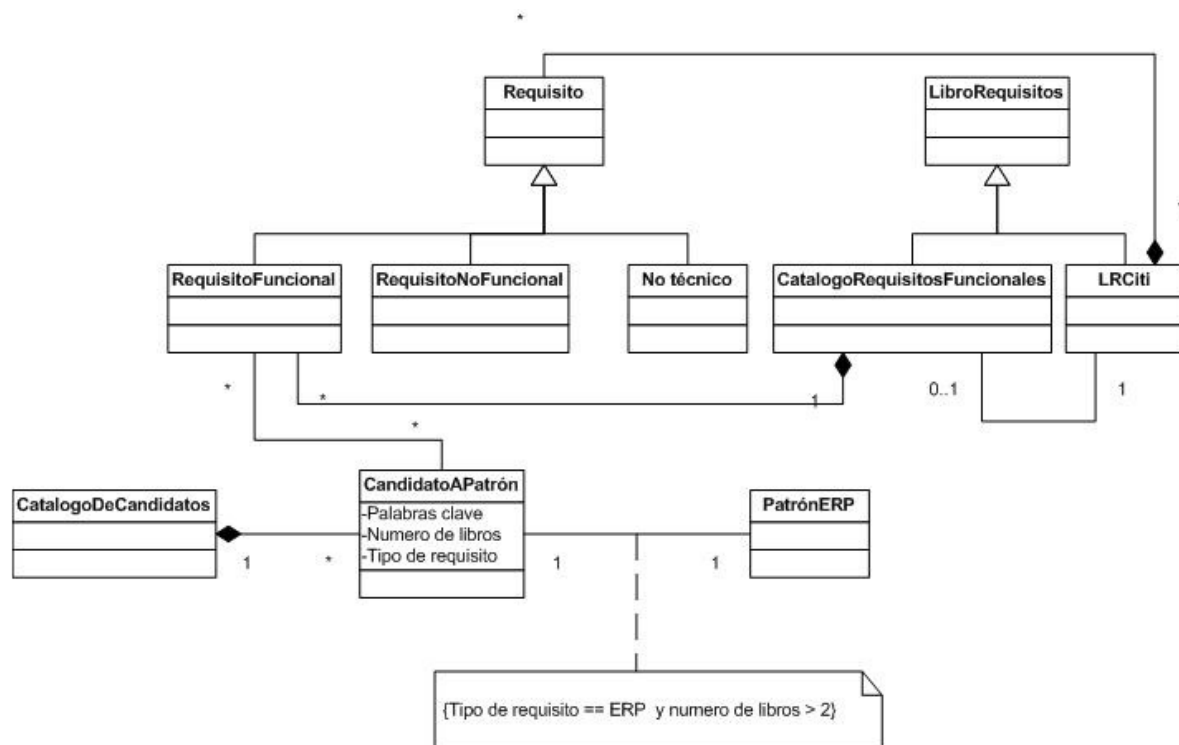


Figura 3-AA: Diagrama UML para la creación de un patrón ERP.

### 3.5 Ejemplo de aplicación del método propuesto.

**Tarea 1. Extracción de requisitos funcionales.** Extraemos de uno de los libros de requisitos el primer requisito, teniendo:

*Definición de los precios de ventas a partir de los precios de compra, márgenes y entregas del proveedor.*

**Tarea 2. Análisis semántico y refinamiento.** A partir del requisito extraído, hacemos el análisis semántica y refinamos (ver figura 3-BB):

Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
Cálculo de precio de ventas (definición de fórmula)	<i>El precio de ventas se calcula en función de ciertos parámetros (los precios de compra, márgenes y entregas del proveedor)</i>	ventas	Precio ventas cálculo Fórmula	Electroauto	4.2.7

Figura 3-BB: Análisis semántico y refinamiento(requisito 1)

**Tarea 3. Inserción en el catálogo de candidatos a patrón.**

Debido a que para este requisito no se han encontrado coincidencias en el catálogo de candidatos, se insertará como un candidato nuevo en dicho catálogo. En este caso tenemos más de un libro de requisitos, por lo que no consideraremos apto a ser patrón a este requisito, hasta haber analizado los requisitos existentes en los otros libros (ver figura 3-CC).

id	#	rel	Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
1	0	0	Cálculo de precio de ventas (definición de fórmula)	<i>El precio de ventas se calcula en función de ciertos parámetros (los precios de compra, márgenes y entregas del proveedor)</i>	ventas	Precio ventas cálculo Fórmula	Electroauto	4.2.7

Figura 3-CC: Catálogo de candidatos a patrón (Candidato nuevo)

**Fin.**

Realizamos una nueva iteración con la finalidad de crear el patrón.

**Tarea 1. Extracción de requisitos funcionales.** Extraemos de uno de los libros de requisitos el primer requisito, teniendo:

*Permitir el cálculo de precios de ventas en función de cambios de algunos parámetros.*

**Tarea 2. Análisis semántico y refinamiento.** A partir del requisito extraído, hacemos el análisis semántica y refinamos (ver figura 3-DD).

Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
Cálculo de precio de ventas (definición de fórmula)	<i>El precio de ventas se calcula en función de cambios en algunos parámetros</i>	ventas	Precio ventas cálculo Fórmula	Mecatech	B.10

**Figura 3-DD: Análisis semántico y refinamiento (requisito 2)**

### Tarea 3. Inserción en el catálogo de candidatos a patrón.

El requisito se inserta como candidato en el catálogo de candidatos a patrones, y se realiza la búsqueda de coincidencias en dicho catálogo. En este caso se han encontrado coincidencias y se han validado, por lo que ha de actualizarse el catálogo de candidatos a patrón aumentando en uno el contador de candidatos asociados e indicando con cuales de ellos se relaciona (ver figura 3-EE).

id	#	rel	Objetivo	Funcionalidad	Módulo	Palabras Clave	Libro	Referencia
1	0	0	Cálculo de precio de ventas (definición de fórmula)	<i>El precio de ventas se calcula en función de ciertos parámetros (los precios de compra, márgenes y entregas del proveedor)</i>	ventas	Precio ventas cálculo Fórmula	Electroauto	4.2.7
2	1	1	Cálculo de precio de ventas (definición de fórmula)	<i>El precio de ventas se calcula en función de cambios en algunos parámetros</i>	ventas	Precio ventas cálculo Fórmula	Mecatech	B.10

**Figura 3-EE: Catálogo de candidatos a patrón (Candidato Asociado)**

### Tarea 4. Inserción en el catálogo de patrones.

Una vez que se ha identificado que existe esta coincidencia en el catálogo de candidatos a patrón y se ha analizado que se trata de un requisito funcional de ERP, se inserta en el catálogo de patrones. En este caso no existe ningún patrón creado con anterioridad por lo que se insertará como un patrón nuevo, formalizándolo.

Finalmente y después de realizar las iteraciones en 6 libros de requisitos obtenemos el patrón de la figura 3-Z.

### 3.6 Aplicaciones

Nuestra idea es que los patrones de requisitos funcionales sean usados:

- En los procesos de selección de los ERP, sobre todo en los primeros pasos. Lo que significa un apoyo en la definición, identificación y establecimiento de necesidades. Ayudando en la validación de las funcionalidades contenidas en los distintos candidatos de ERP.
- Ayudando a la identificación de requisitos funcionales contenidos en el ERP en cuestión y al alineamiento entre requisitos funcionales de la organización y las funcionalidades de los ERP.
- En el apoyo a la creación del call for tenders, ayudando a identificar y priorizar los requisitos funcionales.
- Para la reutilización de requisitos, lo que podría ayudar a generar rápidamente los primeros documentos con todas las características requeridas en un documento en el proceso de elicitación.

Además, los patrones funcionales pueden ser fácilmente utilizados en una herramienta de software para realizar la selección de componentes OTS, en particular de un ERP.

## 4 Conclusiones y líneas futuras

En esta tesis de máster se ha presentado el estado del arte de las propuestas que se ofrecen en las metodologías de selección de componentes OTS y en concreto de ERP detectando los problemas y riesgos que conlleva la mala selección. Además, hemos identificado que no existen muchas propuestas relativas a la mejora de la redacción de requisitos y por tanto de las primeras etapas del proceso de elicitación.

Es por ello que se ha detectado la oportunidad de realizar una contribución mediante la creación de patrones de requisitos funcionales para ERP. Esto con el fin de apoyar la reutilización de requisitos y la mejora en las primeras etapas de la selección de ERP.

En el estado del arte hemos analizado las propuestas sobre patrones de requisitos, detectando que todas hablan de reutilización de conocimiento proponiendo alguna forma de clasificación, sin embargo, en ningún caso se propone un catálogo de patrones para su reutilización.

Para la realización del primer catálogo de patrones funcionales se han analizado 6 libros de requisitos proporcionados por CITI sobre procesos de selección de ERP reales pertenecientes a distintos tipos y tamaños de empresas. Mediante ellos se ha construido un primer patrón con el fin de ejemplificar la creación de dicho catálogo. La utilización de este catálogo permitirá la reutilización de los requisitos funcionales en nuevos proyectos. Reduciendo así el tiempo requerido para la elicitación de requisitos.

Mediante la reutilización de requisitos mediante el uso de patrones se facilita la creación de los libros de requisitos y de call for tenders mejorando la calidad de los mismos. Ayudando en el proceso de selección de un ERP y permitiendo identificar requisitos fácilmente.

Para la realización del catalogo de patrones funcionales se han tomado libros de requisitos de procesos de selección de ERP reales lo cual garantiza la autenticidad de los datos. Su clasificación dentro del modelo ISO mediante los módulos básicos de ERP y el uso de palabras claves facilita la identificación de los requisitos para su posterior reutilización.

Se pretende que este catalogo sea dinámico, que permita:

- La evolución de los requisitos en el tiempo para no perder su validez.
- Ir incluyendo patrones de requisitos de nuevas funcionalidades y extenderlo hacia otros dominios para que estos puedan ser usados en el proceso de selección de software.

### **Líneas futuras**

Durante el proceso de validación de los candidatos a patrones, nos hace falta estudiar mas en profundidad el hecho de si un requisito es propiamente del ERP o de la parametrización.

En comparación con los patrones no funcionales y no técnicos obtenidos por Oscar Méndez hemos detectado que hay una diferencia considerable entre el número de parámetros resultantes en los patrones, ya que en los patrones funcionales son pocos aquellos que distinguen parámetros. Aun no hemos detectado si se debe a la naturaleza propia de los requisitos funcionales o al dominio de los ERP que hemos estudiado, por lo que nos falta estudiarlo en profundidad.

Se pretende ampliar el catalogo de patrones de requisitos funcionales analizando un mayor número de libros de requisitos y publicarlos en un repositorio colaborativo para que podamos irlos evolucionando en conjunto con los miembros de CITI.

Se pretende ampliar el modelo  $i^*$  a un diagrama SR en donde se muestren las dependencias entre las funcionalidades propias de cada modulo básico del ERP.



## 5 Bibliografía

1. Sommerville, I., *Software Engineering* 8. 8 ed. 2006: International Computer Science Series.
2. Prieto-Diaz, R., *Classification of Reusable Modules*, in *Concepts and Models. In software Reusability*, ACM press, Editor. 1989. p. 99-123.
3. Maiden, N., Sutcliffe, A., *Exploring reusable specification through analogy*. Communications of the ACM, 1993. 35 (4): p. 55-64.
4. Barber, k., Graser, T., Grisham, S., Jernigan, S., *Requirements Evolution and Reuse Using the Systems Engineering Process Activities (SEPA)*. Australian Journal of Information Systems (AJIS), 1999. 6 (2).
5. Lam, W., McDermind, J., Vickers, A., *Ten Steps Towards Systematic Requirements Reuse* Third IEEE International Symposium on Requirements Engineering (RE'97) 1997.
6. Finkelstein, A., *Reuse of formatted requirements specifications*. Software Engineering, 1988. 3 (5): p. 186-197.
7. Bolton, D., Jones, S., Till, D., Furber, D., Green, S., *Using Domain Knowledge in Requirements Capture and Formal Specification Construction*. Requirements Engineering: Social and Technical Issues. Academic Press, 1994.
8. Ryan, K., Mathews, B., *Matching Conceptual Graphs as an Aid to Requirements Reus*, I.P.o.t.I.I.S.o.R. Engineering, Editor. 1993. p. 112-120.
9. Miriyala, K., Harandi, T., *Automatic Derivation of Formal Software Specification From Informal Descriptions*. IEEE Transactions on Software Engineering, 1991. 17 (10): p. 1126-1142.
10. Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. 2000: Addison-Wesley.
11. Kolp, M., Giorgini, J., Mylopoulos, *Organizational Patterns for Early Requirements Analysis*. Proceedings 15th CAiSE, 2003.
12. Weiss, M., *Pattern-Driven Design of Agent Systems: Approach and Case Study*. Proceedings 15th CAiSE, 2003, 2003.
13. Booch, G., *Architectural Organizational Patterns*. IEEE Software, 2008. 25(3).
14. Yu, E., *Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering*. Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering, RE'97 1997.
15. Griss, M., Favaro, J., Walton, P., *Managerial and Organizational Issues. Starting and Running a Software Reuse Program*. In *Software Reusability*, 1994: p. 51-78.
16. WISR, *Proceedings 17th Annual Workshop on Software Reuse*, 1995.
17. Hecht, B., *Choose the right ERP software*. 1997, Datamation.
18. Verville, J., Hallington, A., *An investigation of the decision process for selecting an ERP software: the case of ESC*. Management Decision, 2002. 40 No.3: p. 206-216.
19. ILLA, X., Franch, X., Pastor, J., *Formalising ERP Selection Criteria*. Software Specification and Design, 2000. Tenth International Workshop 2000.
20. Rada, R., *Reengineering Software: How to Re-Use Programming to Build New, State of the Art Software*. 1999.

21. López, O., Laguna, M., Marqués, J., *Reutilización del Software a partir de Requisitos Funcionales en el modelo de Mecano: Comparación de Escenarios*. 2001, Universidad de Valladolid.
22. Bhuta, J., Boehm, B., *Attribute-Based COTS Product Interoperability Assessment*. . In ICCBSS '07: Proceedings of the Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems. IEEE Computer Society., 2007: p. 163–171.
23. Galster, M., Eberlein, A., Moussavi, M., *Matching Requirements with Off-the-shelf Components at the Architectural Level*. 6th IEEE International Conference on COTS-based Software Systems (ICCBSS'07), 2007.
24. Madanmohan, T.R., De, R. , *Open Source Reuse in Commercial Firms*. IEEE Software. 21(6) p. 62-69.
25. Stallman, R. *The GNU Operating System and the Free Software Movement*. 1999 [cited; Available from: <http://pascal.case.unibz.it/handle/2038/1589>].
26. FSL. *Free Software Foundation* consultado 2008 [cited; Available from: <http://www.fsfeurope.org/index.es.html>].
27. OSI. *Open Source Initiative*. consultado 2008 [cited; Available from: <http://www.opensource.org/>].
28. Raymond, E. *Adiós "Software Libre"; hola "Código Abierto"*. 1998, última modificación: 2007 [cited; Available from: [http://wiki.kickbill.com/wiki/doku.php?id=goodbye\\_free\\_software\\_hello\\_open\\_source](http://wiki.kickbill.com/wiki/doku.php?id=goodbye_free_software_hello_open_source)].
29. Stallman, R. Overview of the GNU System. 1998, actualizado 2008 [cited; Available from: <http://www.gnu.org/gnu/thegnuproject.html>].
30. Debian.org. *The Debian Free Software Guidelines (DFSG)*. 1997, modificado 2008 [cited; Available from: [http://www.debian.org/social\\_contract.html#guidelines](http://www.debian.org/social_contract.html#guidelines)].
31. OSD. *Open Source Definition*. consultado 2008 [cited; Available from: <http://www.opensource.org/docs/definition.php>].
32. Coar, k. *The Open Source Definition*. 2006 [cited; Available from: <http://opensource.org/docs/osd>].
33. Sommerseth, M., *Component based system development in the norwegian software industry*. . 2006.
34. Raymond, E., *The Cathedral and the Bazaar*. . 1999: O'Reilly Media.
35. Brownsword, L., Oberndorf, T., Sledge C.A., *Developing New Processes for COTS-Based Systems*. IEEE Software, 2000. 17 No.4: p. 48-55.
36. Oberndorf, P., *Facilitating Component-Based Software Engineering: COTS and Open Systems*. Proceedings of the Fifth International Symposium on Assessment of Software Tools- SAST.97, 1997.
37. Vigder, M., Gentleman, M. , Dean, J., , *COTS Software Integration: State of the Art*. 1996.
38. Basili, V.B., B. , *COTS-Based Systems Top 10 List*. IEEE Computer Mayo 2001. 34(5): p. 91-93.
39. Carney D., L.F., *What Do You Mean by COTS? Finally a Useful Answer*. IEEE Software, March/April 2000. 17 (2).
40. Franch, X., Torchiano, M., , *Towards a Reference Framework for COTS-Based Development: A Proposal*. . Proceedings of MPEC'05, St. Louis Missouri, USA. ACM. 2005.

41. Meyers, B., Oberndorf, P., *Managing Software Acquisition*. SEI Series in Software Engineering, 2002.
42. Heineman G., C., W., *Component-based Software Engineering, Putting the Pieces Together*. Addison Wesley, 2001.
43. Crnkovic, I., *Component-based Software Engineering - New Challenges in Software Development*. 25th int. conf. Information Technology Interfaces ITI, 2003: p. 16-19.
44. Brown, A., *Large-scale Component-Based Development*. , ed. P. Hall. 2000.
45. Morisio, M., Reaman, C., Basili, V., Parra, A., Kraft, S., Condon, S., *COTS-Based Software Development: Processes and open issues*. Journal of Systems and Software, 2002. 61: p. 189-199.
46. Crnkovic, I., Larsson, S., *Challenges of Component-Based Development*. Journal of Software System. Elsevier Science Home, April 2002.
47. Ayala, C., Franch, X., *A Goal-Oriented Strategy for Supporting Commercial Off-the-Shelf Components Selection*. Springer: ICSR 2006 LNCS 4039, 2006: p. 1-15.
48. Ruhe, G., *Intelligent Support for Selection of COTS Products*. Lecture Notes in Computer Science, Springer, 2003. 2593: p. 34-45.
49. Ayala, C., *Systematic Construction of Goal-Oriented Taxonomies for Searching and Reusing COTS*. 2008, Universitat politècnica de Catalunya.
50. Ayala, C., Conradi, R. Sørensen, C., Franch, X., *Open Source Collaboration for Fostering Off-The-Shelf Components Selection*. IFIP International Federation for Information Processing, ed. S. Boston. Vol. 234. 2007. 17-30.
51. Alves, C., *Selección de productos de software utilizando ingeniería de requisitos*, in *Informática*. 2001, Universidade Federal de Pernambuco.
52. Prieto-Díaz, R., Feeman, P., *Classifying Software for Reusability*. IEEE Software, Enero 1987.
53. Alves, C., Castro, J. , *A Systematic Method for COTS Selection*. XV Brazilian Symposium on Software Engineering, Rio de Janeiro, Brazil, Octubre 2001.
54. Boehm, B., Abts, C., Bailey, E., *COCOTS Software Integration Cost Model: an Overview*. Proceedings California Software Symposium, Octubre 1998.
55. Yang, Y., Boehm, B., Clark, B., *Assessing COTS Integration Risk Using Cost Estimation Inputs*. Proceedings of the 28th international conference on Software engineering, 2006: p. 431-438
56. Kontio, J., *A Case Study in Applying a Systematic Method for COTS Selection*. Proceedings 18th International Conference on Software Engineering, IEEE Computer Society Press., 1996.
57. Maiden, N., *Acquiring COTS Software Selection Requirements* IEEE Software, Marzo-Abril 1998: p. 46-56.
58. Maiden, N., Kim, H., Ncube, C., *Rethinking Process Guidance for Selecting Software Components*. Proceedings 1st ICCBSS, LNCS 2255, 2002.
59. Phillips, B., Polen, S. , *Add Decision Analysis to your COTS Selection Process*. Software Technology Support Center Crosstalk, 2002.
60. Chung, L., Cooper, K., Courtney, S., *COTS Aware Requirements Engineering and Software Architecting* Proceedings of the SERP, 2004.
61. Dorda, C., Dean, C., Morris, E., Oberndorf, P., *A Process for COTS Software Product Evaluation*. Proceedings of 1st ICCBSS, LNCS 2255, 2002.

62. Ochs, M., Pfahl, D., Chrobok-Diening, G., Nothhelfer-Kolb, B., *A Method for Efficient Measurement-based COTS Assessment and Selection- Method Description and Evaluation Results* Proceedings IEEE 7th International Software Metrics Symposium, 2001: p. 285-296.
63. Kunda, D., *STACE: Social Technical Approach to COTS Software Evaluation*. Component-Based Software Quality - Methods and Techniques, LNCS 2693, 2003.
64. Gregor, S., Hutson, J., Oresky, C., , *Storyboard Process to Assist in Requirements Verification and Adaptation to Capabilities Inherent in COTS*. In Proceedings of the International Conference on COTS-Based Software Systems 2002, 2002.
65. ISO. *Sitio Web de: International Organization for Standarization*. Consultada: Abril 2008 [cited; Available from: <http://www.iso.org/iso/home.htm>.
66. Poston, R., Sexton, M., *Evaluating and Selecting Testing Tools*. IEEE Software 9, 1992. 3: p. 33-42.
67. IEEE, *Recommended Practice on the Selection and Evaluation of CASE Tools (IEEE Std. 1209-1992)*. . New York, NY: Institute of Electrical and Electronics Engineers., 1993.
68. Ayala, C., Franch, X., *Overcoming COTS Marketplace Evolvability and Interoperability*. CAiSE Forum 2006., 2006.
69. Alves, C., Franch, X., Carvallo, J., Finkelstein, A., *Using Goals and Quality Models to Support the Matching Analysis During COTS Selection*. In Lecture Notes in Computer Science (LNCS), 2005. 3412
70. Ayala, C.S., C.; Conradi, R.; Franch, X.; Li, J., *Open Source Collaboration for Fostering Off-The-Shelf Components Selection*. IFIP International Federation of Information Systems, Open Source Systems. International Conference on Open Source Systems, 2007: p. 17-30.
71. Ayala, C., Franch, X., *Domain Analysis for Supporting Commercial-Off-The-Shelf Components Selection*. International Conference on Conceptual Modelling - ER 2006, 2006: p. 354-370.
72. Franch, X., *On the Lightweight Use of Goal-Oriented Models for Software Package Selection*. 17th International Conference on Advanced Information Systems Engineering (CAiSE 2005). , 2005.
73. Navarrete, F., Botella, P., Franch, X., *How Agile COTS Selection Methods are (and can be)?* 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005: p. 160-167.
74. Navarrete, F., Botella, P., Franch, X., *Análisis de los Métodos de Selección de Componentes COTS*. Proceedings of the: Congreso Español de Informática. CEDI'2005. , 2005.
75. Ayala, C., Franch, X., *A Goal-Oriented Strategy for Supporting Commercial Off-The-Shelf Components Selection*. 9th International Conference on Software Reuse., 2006.
76. Carvallo, J., Franch, J., Quer, C., *Managing Non-Technical Requirements in COTS Components Selection*. 14th IEEE International Requirements Engineering Conference (RE'06). 2006.
77. Carvallo, J., Franch, X., *Extending the ISO/IEC 9126-1 Quality Model with Non-Technical Factors for COTS Components Selection*. Workshop on Software Quality (WOSQ'06), 2006.

78. Grau, G., Franch, X., Carvallo, J., Quer, C., *A Software System for Selecting COTS Components*. EUROMICRO'04, 2004.
79. Quer, C., Franch, X., Lopez-Pelegrín, X., *DesCOTS-SL: A Tool for the Selection of COTS Components*. 14th IEEE International Requirements Engineering Conference (RE'06), 2006.
80. Quer, C., Franch, X., Lopez-Pelegrin, X., *DesCOTS-EV: a tool for the evaluation of COTS components*. Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on 2005.
81. Carvallo, J., Franch, J., Grau, G., Quer, C., *On the Use of Quality Models for COTS Evaluation*. MPEC Workshop. Software Engineering. 25rd International Conference ICSE 2004., 2004.
82. Vijay, S., Maiden, N., Franch, X., *Driving Component Selection through Actor-Oriented Models and Use Cases*. Proceedings of the 3rd International Conference on COTS-Based Software Systems (ICCBSS), 2004.
83. Robertson, S., Robertson, J., *Mastering the requirement Process*. 1999: Addison-Wesley.
84. Nuseibeh, B., Easterbrook, S., *Requirements Engineering: A Roadmap*. International Conference on Software Engineering (ICSE2000), 2000: p. 35-46.
85. Sommerville, I., *Integrated Requirements Engineering: A Tutorial*. IEEE Software, 2005: p. 16- 23.
86. Pressman, R., *Ingeniería del Software: Un enfoque práctico*. 5 ed. 2002: McGraw Hill.
87. Toval, A., Nicolás, J., Moros, B., *Un Proceso de Ingeniería de Requisitos basado en Reutilización*. Jornadas de ingeniería de requisitos aplicada, 2001: p. 129-143.
88. Brooks, F., *No Silver Bullet. Essence and accidents of Software Engineering*. IEEE Software, 1987. 20 (4): p. 10-19.
89. McIlroy, D., *Mass produced software components*. Software Engineering Concepts and Techniques. Conference of Software Engineering 1968: p. 88-98.
90. Diaz, R., *Reutilización de Requisitos Funcionales de Sistemas Distribuidos utilizando Técnicas de Descripción Formal in Departamento de Ingeniería Telemática*. 2002, Universidad de Vigo.
91. Krueger, C., *Software Reuse*. ACM Computing Surveys, 1992.
92. Prieto-Díaz, R., *Status Report: Software Reusability*. IEEE Software, 1993. 10(3): p. 61-66.
93. Van Lamsweerde, A., *Requirements engineering in the year 00: A research perspective*. In Proceedings of the 22nd. International Conference on Software Engineering, Limerich. ACM Press, 2000.
94. Bosch, J., *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. 2002: ACM Press. Addison-Wesley.
95. Clements, P., Nothrop, L., *Software Product Lines: Practices and Patterns*. The SEI series in software engineering. Addison-Wesley, 2002.
96. Creps, R.S., M.; Prieto-Díaz, R., *The STARS conceptual framework for reuse processes*. In Proceedings of STARS'92, 1992.
97. Karlsson, E., *Software Reuse: A Holistic Approach* John Wiley & Sons. . 1995.
98. Neighbors, J., *The evolution from software components to domain analysis*. International Journal of Software Engineering and Knowledge Engineering, 1992. 2(3): p. 325-354.

99. Reubenstein, H., Waters, R. , *The Requirements Apprentice: Automated Assistance for Requirements Acquisition*. In IEEE Trans.on Softw Eng., 1991. 17(3): p. 226-240.
100. Cheng, B., Atlee, J., *Research Directions in Requirements Engineering*. IEEE Software, 2007.
101. Clotet, R., Franch, X., Grünbacher, P., López, L., Marco, J., Quintus, M., Seyff, N. , *Requirements Modelling for Multi-Stakeholder Distributed Systems: Challenges and Techniques*  
Proceedings of the 1st International Conference on Research Challenges in Information Science, 2007.
102. Franch, X., *Do We Need Requirements in COTS-Based Software Development?* 3rd International Conference on COTS-Based Software Systems. ICCBSS., 2004.
103. Grünbacher, P., Stallinger, F., Maiden, N.,Franch, X., *A Negotiation-based Framework for Requirements Engineering in Multi-stakeholder Distributed Systems*. Proceedings of the: International Workshop on Requirements Engineering for Open Systems. REOS., 2003.
104. Salazar, G., Botella, P., Dahanajake, A. , *Introducing Non-Functional Requirements in UML*, in *UML and the Unified Process*, L. Favre, Editor. 2003, IRM Press.
105. Durán, A., Bernárdez, B., Ruiz, A., Toro, M., *A Requirements Elicitation Approach Based in Templates and Patterns*. Workshop de Engenharia de Requisitos. (WER99), 1999.
106. Appleton, B. *Patterns and Software: Essential Concepts and Terminology*. 2000. Consultado 2008 [cited.
107. Beck, K., Cunningham, W., *Using Pattern Languages for Object-Oriented Programs*. Submitted to the OOPSLA-87 workshop on the Specification and Design for Object-Oriented Programming, 1987.
108. Coplien, J., *Advanced C++ Programming Styles and Idioms*. 1992: Addison-Wesley.
109. Hagge, L., Lappe, K., *Sharing requirements engineering experience using patterns*. IEEE Software, 2005. 22 (1): p. 24-31.
110. Alexander, C., *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press. 1977.
111. Riehle, D., Züllighoven, H., *Understanding and Using Patterns in Software Development*. Theory and Practice of Object Systems, 1996. 2 (1).
112. Gabriel, R., *Patterns of Software: Tales From the Software Community*. 1998: Oxford University Press.
113. Coplien, J., *Software Patterns* 1996: SIGS.
114. Keepence, B., Mannion, M., Smith, S., *SMARTRe Requirements : Writing Reusable Requirements*. Systems Engineering of Computer Based Systems, 1995., Proceedings of the 1995 International Symposium and Workshop on, 1995: p. 27-34.
115. Tseng, M., Jiao, J., *A variant approach to product definition by recognizing functional requirement patterns*. Elsevier Science Ltd, 1997: p. 629-663.
116. Lu, S., Tcheng, D., *Building layered models to support engineering decision making: A machine learning approach*. Journal of Engineering for Industry ASME Tansactions, 1990: p. 1-9.

117. Mannion, M., Kaindl, H., Wheadon, J., *Reusing Single System Requirements from Application Family Requirements*. In proceedings of the International Conference on Software Engineering, 1999.
118. Fredj, M., Roundies, O., *A Reuse Based Approach for Requirements Engineering* proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, 2001.
119. Fredj, M., Roundies, O., *A Pattern Based Approach for Requirements Engineering*. 10th International Workshop on Database & Expert Systems Applications, 1999: p. 310.
120. Creel, C., *Requirement Patterns*. IEEE Computer Society. Proceedings of the Technology of Object-Oriented Languages and Systems 1999: p. 501.
121. Konrad, S., Cheng, B., *Requirements Patterns for Embedded Systems* 2002.
122. Wahono, R., Cheng, J., *Extensible Requirement Patterns of Web Application for Efficient Web Application Development*. Proceedings of the 1st International Symposium on Cyber Worlds: Theories and Practices (CW2002), 2002.
123. Hagge, L., Lappe, K., Schmidt, T., *REPARE: The Requirements Engineering Patterns Repository*. 13th IEEE International Conference on Requirements Engineering (RE'05), 2005: p. 489-490.
124. Hagge, L., Lappe, K., *Patterns for the RE process*. Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International, 2004: p. 90- 99.
125. Kitchenham, B., Pfleeger, S., *Software Quality: The Elusive Target*. IEEE Software, 1996. 13, No. 1: p. 12-21.
126. Basili, V., Boehm, B., Davis, A., Humphrey, W., Leveson, N., Mead, N., Musa, J., Parnas, D., Pfleger, S., Weyuker, E. , *New Year's Resolution for Software Quality*. Quality Time, J. Hayes Eds. IEEE Software 21, 1 2004: p. 12-13.
127. Voas, J., *Software's Secrets Sauce: The "-ilities [Software Quality]"*. IEEE Software, 2004. 21(6):14.
128. Gillies, A., *Software Quality, Theory and Management*. International Thompson Computer Press, 1997.
129. Chulani, S.S.P.M.D.L.B.D.G., *Deriving a Software Quality View from Customer Satisfaction and Service Data*. European Software Conference on Metrics and Measurement, Marzo, 2001.
130. Chulani, S., Ray, B., Santhanam, P., Leszkowicz, R., *Metrics for Managing Customer View of Quality*. IEEE Metrics conference, 2003.
131. Wong, B., *The Software Evaluation Framework 'SEF' Extended*. Information and Software Technology Journal 2004. 46/15: p. 1037-1047.
132. Wong, B., *Applying the Software Evaluation Framework 'SEF' to the Software Development Life Cycle*. Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE 2003), 2003.
133. Wong, B., Jeffery, R., *Cognitive Structures of Software Evaluation: A Means-End Chain Analysis of Quality*. Proceedings of the Third International Conference, PROFES 2001 2001: p. 6-26.
134. Boehm, B., Huang, L., Jain, A., Madachy, R. , *Quality as Stakeholder Value*. Proceedings of the Second Workshop on Software Quality, ICSE 2004, 2004: p. 1-3.
135. Voas, J., *The Software Quality Certification Triangle*. The Journal of Defense Software Engineering, 1998.

136. Paulk, M., Curtis, B., Chrissis, M., Weber, C., *The Capability Maturity Model for software*, C.M. Software Engineering Institute, Editor. 1993.
137. BOOTSTRAP.team, *BOOTSTRAP: Europe's assessment method*. IEEE Software, 1993.
138. CMMI. *Capability Maturity Model Integration* Consultado 2008 [cited; Available from: <http://www.sei.cmu.edu/cmmi/>].
139. Zinkgraf, S., *Six Sigma: The First 90 Days*. 2006: Prentice Hall.
140. El Emam, K., Drouin, J., Melo, W., *Spice: The Theory and Practice of Soft-ware Process Improvement and Capability Determination*. IEEE Computer Society, 1998.
141. ISO9000, *ISO 9000:2005*. 2005.
142. IEC. *Sitio Web de: the International Electrotechnical Commission*. [cited; Available from: <http://www.iec.ch/>].
143. ISO9126-1, *ISO/IEC 9126-1*. 2001. Ultima revisión 2006.
144. ISO, *ISO/IEC 25000*. 2005.
145. Bøegh, J., *A New Standard for Quality Requirements*. IEEE Computer, 2008: p. 57-63.
146. Botella, P., Burgués, X., Carvallo, J., Franch, X., Grau, G., Marco, J., Quer, C., *ISO/IEC 9126 In Practice: What Do We Need to Know?* Software Measurement European Forum 2004, 2004.
147. Carvallo, J., Franch, J., Quer, C., *Building and Using Quality Models for Complex Software Domains*. <http://citeseer.ist.psu.edu/583315.html>, 2003.
148. Carvallo, J., Franch, J., Quer, C., *Defining a Quality Model for Mail Servers*. COTS-Based Software Systems. Second International Conference, ICCBSS 2003., 2003.
149. Carvallo, J., Franch, J., Quer, C., *Towards a Unified Catalogue of Non-Technical Quality Attributes to Support COTS-Based Systems Lifecycle Activities*. Sixth International Conference on Commercial-off-the-Shelf (COTS)-Based Software System.IEEE Computer Society, 2007.
150. Carvallo, J., Franch, J., Grau, G., Quer, C., *COSTUME: A Method for Building Quality Models for Composite COTS-based Software Systems*. Proceedings of the Fourth International Conference on Quality Software (QSIC'04), 2004: p. 214- 221.
151. Botella, P., Burgués, X., Carvallo, J., Franch, X., Quer, C., *Using Quality Models for Assessing COTS Selection*. Proceedings of the: V Workshop on Requirements Engineering. WER'02, 2002.
152. Burgués, X., Franch, X., *A Language for Stating Component Quality*. XIV Simposio Brasileiro de Engenharia de Software (SBES), 2000.
153. Burgués, X., Franch, X., *Formalising Software Quality Using a Hierarchy of Quality Models*. In Lecture Notes in Computer Science (LNCS), 2004. 0 No. 3180.
154. Burgués, X., Franch, X., Ribó, J., *Una propuesta conforme a MOF para la modelización de la calidad del software*. Proceedings of the: IX Jornadas de Ingeniería del Software y Bases de Datos. JISBD 2004, 2004.
155. Carvallo, J., Franch, J., Grau, G., Quer, C., *Reaching an Agreement on COTS Quality through the Use of Quality Models*. Software Engineering. 25rd International Conference ICSE 2004., 2004.
156. Carvallo, J., Franch, X., *Using Quality Models in Software Component Selection*. IEEE Software, 2003. 20(1).



157. Carvallo, J., Franch, X., *A Quality-Model-Based Approach for Describing and Evaluating Software Components*. Procs. 10th International Conference on Requirements Engineering, 2002.
158. Carvallo, J., Franch, J., Quer, C., Torchiano, M., *Characterization of a Taxonomy for Business Applications and the Relationships among them*. Procs. 3rd International Conference COTS-Based Software Systems, 2004.
159. Botella, P., Burgués, X., Carvallo, J., Franch, X., Pastor, J., Quer, C., *Towards a Quality Model for ERP System Selection*. Component-Based Software Quality. Methods and Techniques., 2003.
160. Duarte, K., Guizzardi, G., de Almeida Falbo, R., *An Ontological Approach to Domain Engineering*. Proceedings of the 14th international conference on Software engineering and knowledge engineering. ACM, 2002: p. 351-358.
161. [SEI-96]. *What is Model Based Software Engineering (MBSE)?* 1996 [cited; Available from: <http://www.sei.cmu.edu/mbse/index.html>.
162. Pigdeon, C., *Organizing and Enabling Domain Engineering to Facilitate Software Maintenance*. 8th Annual Workshop on Software Reuse, IEEE, 1997.
163. Arango, G., *Domain Analysis - From Art Form to Engineering Discipline*. In Proceedings of the 5th International Workshop on Software Specification and Design, CS Press, 1989: p. 152-159.
164. Neighbors, J., *Software Construction Using Components*, in *Department of Information and Computer Science*. Irvine, 1981, University of California.
165. Prieto-Diaz, R., *Domain Analysis: An Introduction*. ACM SIGSOFT Software Engineering Notes 1990: p. 47 - 54.
166. eCOTS. *Software Components Open Directory Project*. consultada 2008 [cited; Available from: <http://www.ecots.org/>.
167. [INCOSE], *International Council on Systems Engineering* 1996-2008. Consultado 2008.
168. IMC, *Internet Mail Consortium*. Consultado 2008.
169. WfMC. *Workflow Management Coalition*. 1993. Consultado 2008 [cited; Available from: <http://www.wfmc.org/>.
170. AIIM. *Enterprise Content Management Association* 2006-2008. Consultado 2008 [cited; Available from: [http://jobs.aiim.org/r/candidates/index.cfm?site\\_id=1633](http://jobs.aiim.org/r/candidates/index.cfm?site_id=1633).
171. Arango, G., Prieto-Diaz, R., *Domain Analysis Concepts and Research Directions in Domain Analysis and Software Systems Modeling*. IEEE Computer Society Press, 1991.
172. Nilson, R., Kogut, P., Jackelen, G. , *Component Provider's and Tool Developer's Handbook Central Archive for Reusable Defense Software (CARDS)*. STARS Informal Technical Report. , U.C.R.V.R.T. Center, Editor. 1994.
173. Van Lamsweerde, A., *Goal-Oriented Requirements Engineering: A Guided Tour*. Proceedings of the 5th IEEE International Symposium on Requirements Engineering, RE'01, Toronto, Canada. 2001.
174. Anton, A.M., M.; Potts, C., *Goal Decomposition and Scenario Analysis in Business Process Reengineering*. Proceedings 6th CAiSE Conference, 1994: p. 94-104.
175. Dardenne, A., Vam Lamswerde, A., Fickas, S., *Goal-Directed Requirements Acquisition*. Science of Computer Programming, 1993: p. 3-50.
176. Habli, I., Wu, W., Attwood, K., Kelly, T., *Extending Argumentation to Goal-Oriented Requirements Engineering*. Advances in Conceptual Modeling –

- Foundations and Applications, ed. L.N.i.C. Science. 2007: Springer Berlin / Heidelberg. 306-316.
177. WEIB, G., *Agent Orientation in Software Engineering, The Knowledge Engineering Review*. Vol. 16(4). 2001: Cambridge University Press. 349 - 373.
  178. Bradshaw, J., Dutfield, S., Benoit, P., Woolley, J., *KAoS: toward an industrial-strength open agent architecture*. 1997: MIT Press. 375 - 418.
  179. Chung, L., Nixon, B., Yu, E., Mylopoulos, J. , *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.
  180. GRL. *Sitio web de GRL*. Fecha de consulta: Mayo 2008. [cited; Available from: <http://www.cs.toronto.edu/km/GRL/>].
  181. D. Amyot, G.M., *URN: Towards a New Standard for the Visual Description of Requirements". Telecommunications and beyond: The Broader Applicability of SDL and MSC*. Third International Workshop (SAM 2002), 2002.
  182. ITU. *International Telecommunication Union*. Consultado: Abril 2008 [cited; Available from: <http://www.itu.int/net/home/index-es.aspx>].
  183. Yu, E., *Modelling Strategic Relationships for Process Reengineering*. . 1995, University of Toronto.
  184. Yu, E., *Understanding 'why' in software process modeling, analysis and design*. In Proceedings of the 16th International Conference on Software Engineering (ICSE'94), 1994: p. 159-168.
  185. TROPOS. *Sitio Web Proyecto TROPOS*. Fecha de consulta: Abril 2008. [cited; Available from: <http://www.troposproject.org>].
  186. Sannicoló, F., Perini, A., Giunchiglia, F., *The Tropos modeling language. a User Guide Technical report DIT-02-0061*. University of Trento, February 2002.
  187. Dubois, E., Du Bois, P., Petit, M., *ALBERT: an agent-oriented language for building and eliciting requirements for real-time systems*. Proceedings of the Twenty-Seventh Hawaii International Conference on 1994. IV: p. 713-722.
  188. Yu, E., Strohmaier, M., Deng, X., *Exploring Intentional Modeling and Analysis for Enterprise Architecture*. Proceedings of the Workshop on Trends in Enterprise Architecture Research (TEAR'06), at the Enterprise Computing Conference (EDOC) 2006.
  189. BRG. *The Business Motivation Model - Business Governance in a Volatile World*. Accesado 2008 [cited; Available from: [www.businessrulesgroup.org](http://www.businessrulesgroup.org)].
  190. Gordijn, J., Yu, E., van der Raadt, B., *e-Service Design Using i\* and e3value Modeling*. IEEE Software, 2006. 3 No. 3: p. 26-33.
  191. Yu, E., Liu, L., Li, Y., *Modelling Strategic Actor Relationships to Support Intellectual Property Management* 20th International Conference on Conceptual Modeling (ER-2001). Spring Verlag, 2001: p. 164-178.
  192. Garzetti, M., Giorgini, P., Mylopoulos, J., Sannicolo, F., *Applying Tropos Methodology to a real case study: Complexity and Criticality analysis*. In Proceeding of the Italian workshop on Dagli OGGETTI agli AGENTI - Dall'informazione alla Conoscenza (WOA02), 2002.
  193. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N., *Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning*. 2004, Informatica e Telecomunicazioni, University of Trento.

194. Liu, L., Yu, E., Mylopoulos, J., *Analyzing Security Requirements As Relationships among Strategic Actors*. in 2nd Symposium on Requirements Engineering for Information Security (SREIS), 2002.
195. Molani, A., Perini, A., Yu, E., Bresciani, P., *Analyzing the Requirements for Knowledge Management using Intentional Analysis*. American Association for Artificial Intelligence, 2003.
196. Sturm, A., Shehory, O., *A Framework for Evaluating Agent-Oriented Methodologies*. Proc. Of AOIS, 2003.
197. UofT. *i\* an agent-oriented modelling framework*. ultima modificacion Diciembre 2004, consulta Abril 2008 [cited; Available from: <http://www.cs.toronto.edu/km/istar/>].
198. i\*Wiki. *i\* Wiki*. 2002–2007 consulta: Marzo 2008 [cited; Available from: [http://istar.rwth-aachen.de/tiki-view\\_articles.php](http://istar.rwth-aachen.de/tiki-view_articles.php)].
199. Grau, G., Cares, C., Franch, X., Navarrete, F., *A Comparative Analysis of i\*Agent-Oriented Modelling Techniques*. Proceedings of The Eighteenth International Conference on Software Engineering and Knowledge Engineering. SEKE06, 2006.
200. Pavon J.; Gomez-Sanz, J., *Agent Oriented Software Engineering with INGENIAS, Proc of CEEMAS 2003, LNAI 2691*. 2003.
201. Dam, K., Winikoff, M., *Comparing Agent-Oriented Methodologies* In the proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information System, 2003.
202. Cernuzzi, L., Rossi, G. , *On the Evaluation of Agent Oriented Modeling Methods*. Proceedings of the 3rd International Conference on Enterprise Information Systems, 2001.
203. Carvallo, J., *Supporting Organizational Induction and Goals Alignment for COTS Components Selection by Means of i\**, in *International Conference on COTS-Based Software Systems (ICCBSS'06)*, I.C.S. Press, Editor. 2006.
204. Ayala, C., Cares, C.; Carvallo, J., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., Quer, C., *Análisis Comparativo de Lenguajes de Modelado Orientados a Objetivos Basados en i\**, in *Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento*. 2004.
205. Ayala, P.C., C.; Carvallo, J.; Grau G.; Haya, M.; Salazar, G.; Franch, X.; Mayol, E.; Quer, C., *A Comparative Analysis of i\*-Based Goal-Oriented Modeling Languages*. Seventeenth International Conference on Software Engineering and Knowledge Engineering. SEKE'05. Howard International House, Taipei, Taiwan. 2005.
206. Franch, X., Grau, G., Quer, C., *Un Marco para la Definición de Métricas sobre Modelos de Dependencias de Actores*. Anais do WER05 - Workshop em Engenharia de Requisitos,, 2005.
207. Franch, X., Haya, M., Mayol, E., *The Use of UML Activity Diagrams and the i\* Language in the Modeling of the Balanced Scorecard Implantation Process*. In *Journal of Computer Science & Technology*, 2005. 5 No.2.
208. Franch, X., Maiden, N., *Modeling Component Dependencies to Inform their Selection*. COTS-Based Software Systems - Second International Conference, ICCBSS 2003, 2003.
209. Cares, C., Franch, X., Mayol, E., *Extending Tropos for a Prolog Implementation: A Case Study Using the Food Collecting Agent Problem*. In *Lecture Notes in Computer Science*. , 2006. 3900.

210. Clotet, R., Franch, X., López, L., Marco, J., Seyff, N., Grünbacher, P., *The Meaning of Inheritance in i\**, in *The 19th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*, T.A. Press, Editor. 2007.
211. López, L., Franch, X., Marco, J., *Defining Inheritance in i\* at the Level of SR Intentional Elements*. In *Proceedings of the 3rd International i\* Workshop istar'08*, 2008: p. 71-74.
212. Grünbacher, P., Dhungana, D., Seyff, N., Quintus, M.I., Clotet, R., Franch, X., López, L., Marco, J., *Goal and Variability Modeling for Service-oriented System: Integrating i\* with Decision Models*, in *Software and Service Variability Management Workshop - Concepts, Models, and Tools*, E.N. Tomi Männistö, Mikko Raatikainen, Editor. 2007, Tomi Männistö, Eila Niemelä, Mikko Raatikainen.
213. Grau, G., Franch, X., Mayol, E., Ayala, C., Cares, C., Haya, M., Navarrete, F., Botella, P., Quer, C., *RiSD: A Methodology for Building i\* Strategic Dependency Models*. *Proceedings of the: Seventeenth International Conference on Software Engineering and Knowledge Engineering. SEKE'05*. Howard International House, 2005.
214. Grau, G., Franch, X., Ávila, S., *J-PRiM: A Java Tool for a Process Reengineering i\* Methodology*, in *Proceedings of the 14th IEEE International Requirements Engineering Conference*. 2006.
215. Franch, X., Grau, G., Mayol, E., Quer, C., Ayala, C., Cares, C., Navarrete, F., Haya, M., Botella, P., *Systematic Construction of i\* Strategic Dependency Models for Socio-Technical Systems*. In *International Journal of Software Engineering and Knowledge Engineering*, 2007. 17 No. 1.
216. Grau, G., Franch, X., Maiden, N., *REDEPEND-REACT: an Architecture Analysis Tool*. 3th IEEE Requirements Engineering International Conference (RE'05). IEEE Computer Society, 2005.
217. Pages, C., de-Marcos, L., Martínez, J., Gutiérrez, J. , *Definición de Métricas de Calidad en el Proceso de Parametrización de sistemas ERP*. RPM-AEMES, 2007. 4 No. 3: p. 74-81.
218. Markus, M., Cornelius, T. , *The Enterprise systems experience. From adoption to success*. In R.W. Zmud, Ed., *Framing the Domains of IT Research: Glimpsing the Future Through the Past*. Pinnaflex Educational Resources, 2000.
219. Davenport, T., *Putting the enterprise into the enterprise system*. Harvard University Graduate School of Business Administration 1998. 76, (4) p. 121 - 131.
220. Dean, J., Gravel, A. , *Enterprise Resource Planning: Global Opportunities and Challenges*. 2002: Springer.
221. Klaus, H., Rosemann, M., Gable, G. , *What is ERP?* information systems frontiers, 2000. 2 (2): p. 141-162.
222. Kalakota, R., Robinson, M., *e-Business 2.0: Roadmap to Success*, ed. M.A.-W. Reading. 2001.
223. Møller, C., *ERP II: a conceptual framework for next-generation enterprise systems?* Journal of Enterprise Information Management, 2005. 18 No. 4: p. 483-497.
224. Bond, B., Genovese, Y., Miklovic, D., Wood, N., Zrimsek, B., Rayner, N., *ERP Is Dead — Long Live ERP II*. GartnerGroup, 2000.

225. IDC-Press. *European ERP Applications Market Back to Healthy Growth as New Trends Kick in, Says IDC* consultado 2008 [cited].
226. Jacobson, S., Shepherd, J., D'Aquila, M., Carter, K. *The ERP Market Sizing Report, 2006–2011*. 2007 [cited].
227. Lucas, K. *La Situación de la Adopción de Software de Gestión Empresarial en Europa*. 2007 [cited].
228. CB-Consulting. *Informe sobre el Mercado de las Soluciones de Gestión Empresarial en España 2007*. 2007 [cited].
229. Gore, A., *Exploring the Competitive Advantage Through ERP Systems: From Implementation to Applications in Agile Networks*, in *Industrial Engineering and Management*. 2008, Universidad de OULU.
230. Esteves, J., Pastor, J., *An ERP Life-cycle- based Research Agenda*. First International workshop in Enterprise Management and Resource Planning: Methods, Tools and Architectures – EMRPS'99, Venice, Italy 1999.
231. Pnina, S.B., G.; Dov, D., *Aligning an ERP system with enterprise requirements: an object-process based approach*. Elsevier, 2005: p. 639-662.
232. Koch, C., *BPR and ERP: realising a vision of process with IT*. Business Process Management Journal, 2001: p. 258-265.
233. Gibson, N., Holland, C., Light, B., *Enterprise resource planning: A Business Approach to Systems Development* in proceedings of the 32nd Hawaii International Conference on Systems Sciences, 1999.
234. Sieso, D., Agüero, A., *ERP en Cifras*.
235. Soffer, P., Golany, B., Dori, D., *Aligning an ERP system with enterprise requirements: an object-process based approach*. Elsevier, 2005: p. 639-662.
236. Chen, I., *Planning for ERP systems: Analysis and future trend*. Business Process Management Journal, 2001. 7(5): p. 374-386.
237. Puig, J. *Guía del Software Empresarial - Los ERP Y SAP R/3 como Sistemas de Información empresariales*. consultada 2008 [cited; Available from: <http://paginespersonals.upcnet.es/~jpi2/erp/temas/erp.html>].
238. Fui-Hoon Nah, F., *Enterprise Resource Planning Solutions and Management*. 2002: Idea Group Inc (IGI).
239. Gray C.D and Landvater, D.V., *MRPII Standard Systems*. Oliver Wight publications Limited, Brattleboro, VT., 1989.
240. Møller, C., *ERP II - Next-generation Extended Enterprise Resource Planning*. Proceedings of the Seventh World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, US. 2003.
241. Jawadekar, W., *Management Information Systems Text and Cases*. 3ed ed. 2007: Tata McGraw-Hill.
242. Kalakota, R., Robinson, M. , *E-Business: Roadmap for success*. 1999: Addison Wesley, Massachusetts.
243. Capron, B., Kuiper, D. , *Are You Ready For a New System?* Manufacturing Systems, 1998: p. A18-A20.
244. Pastor, J., Franch, X., Sistach, F., *Methodological ERP Acquisition: the SHERPA Experience*. Segunda ed, ed. W.C.I.S.M. Guide. 2002: McGraw-Hill.
245. Estay, C., Pastor, J., *Selección de ERP en Pequeñas y Medianas Empresas con un Proyecto de Investigación-Acción*. 1er. Workshop en: Métodos de Investigación y

- Fundamentos Filosóficos en Ingeniería del Software y Sistemas de Información (MIFISIS'2002) 2002.
246. Rolland, C., Prakash, N., *Matching ERP System Functionality to Customer Requirements*. IEEE Computer, 2001: p. 66-75.
  247. Chiesa, F., *Metodología para Selección de Sistemas ERP*. 2004, Instituto Tecnológico de Buenos Aires. p. 17-37.
  248. Wei, C., Chien, C., Wang, M., *An AHP-based approach to ERP system selection*. International Journal of Production Economics, Abril 2005. 96 (1): p. 47-62.
  249. Lien, C., Chan, H., *A Selection Model for ERP System by Applying Fuzzy AHP Approach*. International Journal of the computer, the internet and management, 2007: p. 58-72.
  250. Cil, I., Alpturk, O., Yazgan, H., *A new collaborative system framework based on a multiple perspective approach: InteliTeam*. Decision Support Systems, 2005. 39(4): p. 619-641.
  251. Verville J., H.A., *A six-stage model of the buying process for ERP software*. Science Direct, 2002: p. 10.
  252. Teltumbde, A., *A framework of evaluating ERP projects*. International Journal of Production Research 38, 2000: p. 4507–4520.
  253. Liao, X., Li, Y., Lu, B., *A model for selecting an ERP system based on linguistic information processing*. Information Systems, 2007. 32(7): p. 1005-1017.
  254. Stefanou, C., *A framework for the ex-ante evaluation of ERP software*. European Journal of Information Systems, 2001: p. 204-215.
  255. Shakir, M., *Decision Making in the Evaluation, Selection and Implementation of ERP Systems*. in Proceedings of the 6th Americas Conference on Information Systems, 2000: p. 1033-1038.
  256. Sammon, D., Adam, F. , *Toward a Model for Investigating Non-Decision Making in ERP Communities*. in Proceedings of the International Conference on Decision Making and Decision Support in the Internet Age, IFIP TC8/WG8.3, 2002.
  257. Sammon, D., McAvinue, D. , *Investigating Non-Decision Making during an ERP Software Selection Process*. IFIP TC8/WG 8.3, International Conference., 2004.
  258. Pastor, J., Franch, X., *On the Formalisation of ERP Systems Procurement*. Proceedings of the: Second Workshop on Commercial Off-The-Shelf Software, 2000.
  259. Pastor, J.F., X.; Sistach, F., *Methodological ERP acquisitions: the SHERPA experience*, in *The Guide to IT Service Management*., A.-W. Jan van Bon, Editor.
  260. Burgués, X., Franch, X., Estay, C., Pastor, J., Quer, C., *Combined Selection of COTS Components*. COTS-Based Software Systems: First International Conference, ICCBSS02., 2002.
  261. Esteves, J., Pastor, J., Casanovas, J., *Organizational and National Issues of an ERP Implementation in a Portuguese Company*. Proceedings of the IFIP Conference (w 8.2 + w9.4). , 2003.
  262. Davis, A., *201 Principles of Software Development*, ed. McGraw-Hill. 1995.
  263. Mendez, O., Franch, X., Quer, C., *Requirements Patterns for COTS Systems* Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), 2008: p. 232-234.
  264. Decreus, K., Poels, G., *Putting Business into Business Process Models*. 2008, Faculty of Economics and Business Administration, Ghent University.

265. M. Krystkowiak, V.B., E. Dubois, *Efficient COTS selection with OPAL tool*. Proceedings of International Workshop on Models and Processes for the Evaluation of COTS Components - MPEC'04, Edinbourg, 2004.

## 6 Apéndice 1

A continuación se muestran los módulos pertenecientes a cada ERP utilizado para obtener los módulos básicos

### Oracle JDEduards

<a href="http://www.oracle.com/global/es/products/applications/jdedwards-enterprise-one.html">http://www.oracle.com/global/es/products/applications/jdedwards-enterprise-one.html</a>									
<b>JDEduards</b>									
Assets lifecycle management	Capital Asset Management	Condition-Based Maintenance	Equipment Cost Analysis	Preventive Maintenance					
Customer Relationship Management	Advanced Pricing	Branch Scripting	Case Management	Customer Self Service	PIM Sync	Sales Order Management	Service Management	Sales Force Automation	Solution Advisor
Financial management	Accounts Payable	Accounts Receivable	Advanced Cost Accounting	Expense Management	Fixed Asset Accounting	Financial Management and Compliance Console	General Ledger		
Human capital management	Australia/New Zealand Payroll	Canadian Payroll	Human Resources Management	U.S. Payroll	Employee Self Service	Manager Self Service	Time and Labor	eRecruit	
Project Management	Contract and Service Billing	Project Costing	Project and Government Contract Accounting						
Supply chain management	Food and Beverage Producers	Supply Chain Execution (Logistics)	Supply Management (Procurement)	Manufacturing	Supply Chain Planning (SCP)				
Supply Management (Procurement)	Agreement Management	Procurement and Subcontract Management	Requisition Self Service	Buyer Workspace	Product Variants	Supplier Self Service	Operational Sourcing		
Tools and technologies	Oracle Technology Foundation	IBM Technology Foundation	BI	Server Manager	Business Services				



## Oracle PeopleSoft

<http://www.oracle.com/lang/es/applications/suites.html#PSFT>

PeopleSoft										
Campus Solutions	Academic Advisement	Financial Aid	Student Administration	Campus Community	Gradebook	Student Financials	Campus Self Service	Recruiting and Admissions	Student Records	Contributor Relations
Human Capital Management	Assess-Design-Develop	Plan-Attract-Onboard	Workforce Performance Solutions	Optimize-Track-Monitor	Plan-Incent-Reward	Workforce Scheduling				
Customer Relationship Management	CRM Analytics	Marketing Solution	Sales Solution	CRM Industry Solutions	Partner Relationship Management Solution	Service Solution				
Service Automation	Project Portfolio Management	Program Management	Resource Management	Proposal Management	Project Billing	Project Costing	Grants	Mobile Time and Expense	Contracts	Expenses
Enterprise Performance Management	CRM Analytics	Industry Analytics	Supply Chain Analytics	Financial Analytics	Performance Management Warehouse	Workforce Analytics				
Supplier Relationship Management	Catalog Management	Purchasing	Supplier Contract Management	Collaborative Supply Management	Services Procurement	Supplier Rating System	eProcurement	Strategic Sourcing	Supply Chain Warehouse	eSupplier Connection
Supply Chain Management	Customer Order Management	Inventory and Fulfillment Management	Manufacturing Solution	Supply Chain Planning						
Financial Management	Asset Lifecycle Management	Financial Management	Treasury	Financial Analytics	Profitability Management for Financial Services	Government Portal				

## Oracle

<http://www.oracle.com/lang/es/applications/suites.html>

ORACLE E-BUSINESS SUITE						
Advanced Procurement	iProcurement	Purchasing	Sourcing	iSupplier Portal	Oracle Supplier	Daily Business
Contracts	Oracle Procurement	Oracle Project	Oracle Sales	Oracle Service		
Corporate Performance Management	Business Intelligence	Enterprise Planning	Profitability	Balanced Scorecard	DBI (Daily	Financial
Customer Data Management						
Customer Relationship Management	Oracle Channel	Oracle Order	Oracle service	Oracle Marketing	Oracle Sales	
Financials	Asset Lifecycle	Financial Control &	Procure-To-Pay	Cash & Treasury	Financial	Travel & Expense
Human Resources Management	Incentive Compensation	Human Resources	Performance	Workforce	Compensation	Payroll
Intelligence	Daily Business Intelligence	Daily Business	Daily Business	Daily Business		
Interaction Center	Advanced Inbound	Email Center	Scripting	Advanced		
Learning Management						
Logistics	Inventory Management	Transportation	Warehouse	RFID		
Maintenance	Complex Maintenance,	Enterprise Asset				
Manufacturing	Discrete Manufacturing	Oracle Mobile Supply	Production	Flow Manufacturing	Process	Shop Floor
Marketing	Marketing	Trade Management				
Order Management	Oracle Advanced Pricing	Global Order	Configurator	Order Management	Deal	Oracle Mobile Supply
Product Lifecycle Management	AutoVue 2D Professional	AutoVue Electro-	AutoVue VueLink	AutoVue 3D	AutoVue Office	AutoVue Web Version
Projects	Project Portfolio Analysis	Project Management	Project Resource	Daily Business	Project	Project Contracts
Sales	Incentive Compensation	Proposals	Sales Contracts	iStore	Quoting	Sales for Handhelds
Service	Depot Repair	Interaction Center	iSupport	Field Service	Advanced	Service Contracts
Supply Chain Execution	Order Management	RFID (Radio	Transportation		Warehouse	
Supply Chain Management						
Supply Chain Planning	Advanced Supply Chain	Global Order	Real-Time Sales	Collaborative	Inventory	Strategic Network

Continuación

Procurement Contracts	Services					
Credit-To-Cash	Governance, Risk					
Tutor	Daily Business	Learning	Time and	Advanced	iRecruitmen	Self Service
Transportation	Warehouse					
AutoVue EDA						
Project Costing	Project Billing	Time and				
Partner Management	Sales	TeleSales				
Advanced Scheduler	Advanced	Mobile Field	Email	Spares	Scripting	TeleService
Demand Management	Production					

## SAP

<http://www.sap.com/solutions/businessmaps/78590453EC454B8986A933EBB7E84848/index.epx>

SAP							
Analytics	Strategic Enterprise Management	Financial Analytics	Operations Analytics	Workforce Analytics			
Financials	Financial Supply Chain Management	Tesoreria	Financial Accounting	Management Accounting	Corporate Governance		
Human Capital Management	Talent Management	Workforce Process Management	Workforce Deployment				
Procurement and Logistics Execution	Procurement	Inventory and Warehouse Management	Inbound and Outbound Logistics	Transportation Management			
Product Development and Manufacturing	Production Planning	Manufacturing Execution	Product Development	Life-Cycle Data Management			
Sales and Service	Sales Order Management	Aftermarket Sales and Service	Professional-Service Delivery				
Corporate Services	Real Estate Management	Enterprise Asset Management	Project and Portfolio Management	Travel Management	Environment, Health and Safety Compliance Management	Quality Management	Global Trade

## Microsoft Axapta

<a href="http://www.microsoft.com/dynamics/ax/using/productdocumentation/users/default.aspx">http://www.microsoft.com/dynamics/ax/using/productdocumentation/users/default.aspx</a>								
<b>Axapta - Microsoft AX</b>								
<b>Accounting</b>	Chart of accounts	Budgets	Fixed assets	Dimensions	Dimension sets and dimension set hierarchies	Financial statements	Journals and posting	Managing ledger accounts
<b>Cost accounting</b>	Setting up cost accounting	Cost categories	Dimensions in cost accounting	Journals and posting in cost accounting	Structures	Expense distribution sheet	Calculations, allocations, and distributions	Budgeting in cost accounting
<b>Bank</b>	Make a payment by check	Set up a method of payment for checks	Delete checks (class form)	Create a deposit slip	Void unposted checks	Generate a check on a ledger account or a bank account	Reconcile a bank account	bank groups
<b>Accounts payable</b>	Purchase orders in general	Set up accounts payable	Purchase blanket orders and release orders	Delivery addresses for purchase orders	Miscellaneous charges for purchase orders	Status of purchase orders	Post purchase orders	Supplementary items for purchase orders
<b>sales and marketing</b>	Contact person list	Business relations	Import business relations	Campaigns	Mailings	Telemarketing	Activities	Quotation
<b>Accounts receivable</b>	Sales orders in general	Quotation workflow	Accounts receivable setup	Sales blanket and release orders	Delivery dates for sales orders	Delivery addresses for sales orders	Direct deliveries	Free text invoices
<b>Inventories</b>	Inventory items in general	Setting up inventory management	Item dimensions	Inventory dimensions	Item groups	Product groups	Quarantine orders	Item backorders
<b>PRODUCTION</b>	Common production principles	Production setup	Create productions	Estimate production	Schedule production	Release production orders	Start production orders	Report production orders as finished
<b>plan provisions</b>	Master planning concepts	Master planning setup	Master planning scheduling	Master planning results				
<b>control floor</b>								
<b>HR</b>	Organization units	Positions	Employees in human resources	Development plans	Absence	Recruitment	Courses	Actions
<b>projects</b>	Invoicing	Using statistics	Adjusting transactions	Generating and managing periods	Setting up cost prices and sales prices	Calculating the cost to complete	Manage employees	Setting up estimates
<b>Questionnaires</b>	Creating questionnaires	Working with questions	Working with answers	Completing questionnaires	Distributing questionnaires	Evaluating results	Setting-up questionnaire parameters	
<b>Scorecard</b>								
<b>services</b>	Service agreements	Subscriptions	Time windows	Repair management	Reason codes	Service objects and object groups	Service templates	Service orders
<b>manufacturing</b>	Product Builder Introduction	Product Builder key processes	Concepts and data flow	Preliminary setup	Configurable items	Item modeling variables	Customer specific default values	Product models

## Continuación

Cash flow forecasts and currency requirements	Month, period, and fiscal year closing	Fiscal year closing checklist	Consolidations	Sales tax	Withholding tax	Company currency conversion							
Periodic activities in cost accounting													
bank transaction types	bank accounts	check layouts											
purchase orders.	Purchase backorders	Packing material for purchase orders	Intercompany purchase orders	Invoice journals	Vendor payments	Management of transactions	Exchange adjustment of open vendor transactions	Promissory notes	United States tax 1099				
E-mails	Document management	Encyclopedia	Sales management statistics	Price simulation									
Intrastat for sales orders	Supplementary items for sales orders	Miscellaneous charges for sales orders	Packing material for sales orders	Sales backorders	Post sales orders	Status of sales orders	Intercompany sales orders	Customer payments	Overdue customer payments	exchange adjustment of open customer transactions	Bills of exchange	Letras de cambio	
Inventory closing	Mark inventory	Manage bills of materials (BOM)	Warehouse management	RFID	Transfer orders	Reserve inventory	Packing materials and packing	Italian fiscal LIFO	Inventory journals	Supplementary items	Post inventory	Status of inventory	
End production													
Strategic plans	Human resources statistics	Human resources parameters											
Quotation	Setting up the Work In Process (WIP) structure	Creating and enabling validation	Creating and maintaining project groups	Posting	Setting up line properties	Creating and applying approval procedures	Creating and maintaining journals	Creating and maintaining categories	Creating cost templates	Creating and using credit notes	Quotation (Project)	Posting fixed-price projects examples	Creating and managing transactions
Bills of materials (BOM).	Service tasks	Service order stages											
Validation rules	Testing the product model's user dialog box	Set up price combinations	Modeling tree	Visualization	Compile the product model	Test the product model	Wizard for product model creation	Configuring items					

## Microsoft Nav

<http://www.microsoft.com/spain/businesssolutions/dynamics/productos/nav/default.mspix>

MicrosoftNAV - NAVISION				
Gestión financiera	CONTABILIDAD	Cobros	ACTIVOS FIJOS	PAGOS
Análisis				
suministros	Almacén	Gestión de inventario	Fabricación	Planificación del Suministro
Fabricación	Fabricación Básica	Fabricación Ágil	Planificación de suministro	Planificación de capacidad
Distribución	Gestión de Almacén	Gestión de inventario		
Gestión de recursos humanos				
Gestión de proyectos	Proyectos	Recursos		
Área básica				
ventas y marketing				
gestion de servicios				
Business Intelligence				
Portal de Empleado				

**ORCHESTRA**

<http://www.alizee-info.fr/solutions.htm>

gestión de inventario  
Configurador  
Gestión de previsiones  
Módulo de clasificación funcional  
Plan Maestro - el cálculo de las necesidades  
Administración de ventas  
Gestión de fabricación  
General de Contabilidad  
Gestión de la subcontratación  
Contabilidad clientes  
gestión de herramientas  
Contabilidad proveedores  
Seguimiento de actividades en tiempo real  
Contabilidad analítica  
Gestión de gastos  
Los cuadros de gestión y los presupuestos  
Compra de gestión  
Gestión de perspectivas y estimaciones

**pmiX**

<http://www.pmigest.be/fr/pages/erp-fonctions.aspx>

Inventario y logística  
Ventas  
Compras  
e-commerce  
Production  
Oferta  
Servicios  
Control de calidad  
Multisitios  
Documentos  
Time-sheet  
Marketing  
Finanzas

**IFS**

<http://www.ifsworld.com/solutions2/default.asp>

Assets  
Business Modeling  
Corporate Performance  
Customers  
Documents  
Financials  
Human Resource  
Maintenance, Repair & Overhaul  
Manufacturing  
Product Lifecycles  
Project Centric Businesses  
Quality Management  
Service Management  
Supply Chain Management  
Value Chain Collaboration



## 7 Apéndice 2

A continuación se presentan algunos de los patrones obtenidos hasta el día de hoy

<b>Identificador: 2 Nombre:</b> Generación de previsiones por calendario			
<b>Módulo</b>	Finanzas		
<b>Palabras clave</b>	Calendario, previsión, cobro, pago		
<b>Versión</b>	1 (marzo 2008)		
<b>Autor</b>	GESSI (UPC)		
<b>Fuente</b>	Libro de requisitos proporcionado por CITI		
<b>Objetivo del Patrón</b>	Generar las previsiones en función de un calendario		
<b>Descripción</b>	El patrón expresa la necesidad de generar previsiones de pago, cobro, etc., siguiendo un calendario por periodo. Se toma en cuenta que el periodo puede ser calculado en base a plazos de entrega, etc.		
<b>Parte fija del patrón</b>	<i>Un calendario por periodo debe generar las previsiones de x</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	<i>X</i>	<i>conjunto no vacío de Operaciones Financieras</i>	<i>conjunto de Operaciones Financieras</i>
		<i>Operaciones financieras</i>	<i>Nominal {pago, cobro}</i>
<b>Extensión</b>	<b>Identificador: 2.1 Nombre:</b>		
	<i>El periodo debe ser calculado de acuerdo a un plazo de entrega</i>		
<b>Comentarios</b>	<comentarios>		

<b>Identificador: 3 Nombre:</b> Catálogo de productos y servicios con información estandarizada			
<b>Módulo</b>	Ventas		
<b>Palabras clave</b>	Catálogo, servicios, información, estándar		
<b>Versión</b>	1 (marzo 2008)		
<b>Autor</b>	GESSI-UPC		
<b>Fuente</b>	Libro de requisitos proporcionado por CITI		
<b>Objetivo del Patrón</b>	Crear un catálogo estandarizado de productos y servicios		
<b>Descripción</b>	El patrón expresa la necesidad de poseer un catálogo de productos y servicios con información estandarizada. Tomando en cuenta el permitir añadir, modificar y eliminar campos y sus correspondientes reglas de contenido de dichos catálogos		
<b>Parte fija del patrón</b>	<i>El catálogo de productos y servicios debe contener información estandarizada</i>		
<b>Extensión</b>	<b>Identificador: 3.1 Nombre:</b> ABM de campos y reglas de contenido		
	<i>El sistema debe permitir añadir, modificar y eliminar campos de información con sus reglas de contenido</i>		
<b>Comentarios</b>	<comentarios>		

<b>Identificador: 4 Nombre:</b> Reducción en precio de venta	
<b>Módulo</b>	Ventas
<b>Palabras clave</b>	Reducción, precio de venta, promoción
<b>Versión</b>	1 (marzo 2008)
<b>Autor</b>	GESSE-UPC
<b>Fuente</b>	Libro de requisitos proporcionado por CITI
<b>Objetivo del Patrón</b>	Aplicar promociones al precio de venta
<b>Descripción</b>	El patrón expresa la necesidad de aplicar descuentos al precio de venta de los productos. Se toma en cuenta el poder aplicarlo a un solo producto o a un conjunto de ellos, a un cliente o a un conjunto de clientes, el periodo de promoción, etc.
<b>Parte fija del patrón</b>	<i>El sistema debe aplicar promociones al precio de venta de los productos</i>
<b>Extensión</b>	<b>Identificador: 4.1 Nombre:</b> Aplicación de reducción en precio de ventas
	<i>La reducción al precio de ventas debe aplicarse a un producto o a un conjunto de productos, a un cliente o a un conjunto de clientes en un determinado periodo de tiempo</i>
<b>Comentarios</b>	<comentarios>

<b>Identificador: 5 Nombre:</b> Asociación de precios para un mismo producto			
<b>Módulo</b>	Finanzas		
<b>Palabras clave</b>	Precio, producto		
<b>Versión</b>	1 (marzo 2008)		
<b>Autor</b>	GESSE-UPC		
<b>Fuente</b>	Libro de requisitos proporcionado por CITI		
<b>Objetivo del Patrón</b>	Asociar distintos precios a un mismo producto		
<b>Descripción</b>	El patrón expresa la necesidad de asociar distintos precios a un mismo producto, tal como el precio de ventas, el precio de compra, el precio de producción, etc.		
<b>Parte fija del patrón</b>	<i>Un producto debe tener varios precios</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	<i>Precios</i>	<i>Conjunto de P</i>	<i>Conjunto de P</i>
		<i>P</i>	<i>Nominal {Precio de venta, precio de compra, precio especial, precio de descuento, precio de producción, etc.}</i>
<b>Comentarios</b>	<comentarios>		

<b>Identificador: 6 Nombre:</b> Cálculo de precio de coste			
<b>Módulo</b>	Finanzas		
<b>Palabras clave</b>	Cálculo, coste, precio		
<b>Versión</b>	1 (marzo 2008)		
<b>Autor</b>	GESSI-UPC		
<b>Fuente</b>	Libro de requisitos proporcionado por CITI		
<b>Objetivo del Patrón</b>	Calcular el precio de coste de un producto		
<b>Descripción</b>	El patrón expresa la necesidad de determinar el precio de coste del producto o servicio. El precio es determinado dependiendo de ciertos factores como materia prima, proveedor, depreciación de la maquinaria, alquiler, energía, gastos de transportación, etc.		
<b>Parte fija del patrón</b>	<i>El cálculo del precio de coste debe realizarse usando P</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	<i>P</i>	<i>conjunto de factores</i>	<i>conjunto de factores</i>
		<i>Factores</i>	<i>Nominal {proveedor, depreciación de maquinaria, energía, alquiler, coste de transportación, materia prima, etc.}</i>
<b>Extensión</b>	<b>Identificador: 6.1 Nombre:</b> Actualización de precio de coste		
	<i>El precio de coste debe ser actualizado automáticamente cuando ocurre un cambio en P</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	<i>P</i>	<i>conjunto de factores</i>	<i>conjunto de factores</i>
		<i>Factores</i>	<i>Nominal {proveedor, depreciación de maquinaria, energía, alquiler, coste de transportación, materia prima, etc.}</i>
<b>Comentarios</b>	<comentarios>		

Identificador: 7 Nombre: Pagos múltiples a una misma factura	
Módulo	Finanzas
Palabras clave	Pago, factura
Versión	1 (marzo 2008)
Autor	GESSE-UPC
Fuente	Libro de requisitos proporcionado por CITI
Objetivo del Patrón	Permitir pagos múltiples a una misma factura
Descripción	El patrón expresa la necesidad de determinar el precio de coste del producto o servicio. El precio es determinado dependiendo de ciertos factores como materia prima, proveedor, depreciación de la maquinaria, alquiler, energía, gastos de transportación, etc.
Parte fija del patrón	<i>El pago de una factura debe poder realizarse en forma total o parcial de acuerdo a ciertas condiciones</i>
Comentarios	<comentarios>

Identificador: 8 Nombre: Creación de reportes de facturación			
Módulo	Finanzas		
Palabras clave	Reporte, factura		
Versión	1 (marzo 2008)		
Autor	GESSE-UPC		
Fuente	Libro de requisitos proporcionado por CITI		
Objetivo del Patrón	Generar reporte de facturación		
Descripción	El patrón expresa la necesidad de generar un reporte de facturación en base a un conjunto de entradas		
Parte fija del patrón	<i>El reporte de facturación debe generarse utilizando x como entrada</i>		
	Nombre del parámetro	Nombre de la métrica	Tipo de métrica
	x	conjunto de criterios	conjunto de criterios
		criterios	Nominal {proveedor, fecha de pago, fecha de generación/recepción, fecha de pago/cobro, etc.}
Comentarios	<comentarios>		

<b>Identificador: 9 Nombre:</b> Cancelación de Factura			
<b>Módulo</b>	Finanzas		
<b>Palabras clave</b>	Cancelación, factura		
<b>Versión</b>	1 (marzo 2008)		
<b>Autor</b>	GESSI-UPC		
<b>Fuente</b>	Libro de requisitos proporcionado por CITI		
<b>Objetivo del Patrón</b>	Cancelar facturas emitidas		
<b>Descripción</b>	El patrón expresa la necesidad de determinar el precio de coste del producto o servicio. El precio es determinado dependiendo de ciertos factores como materia prima, proveedor, depreciación de la maquinaria, alquiler, energía, gastos de transportación, etc.		
<b>Parte fija del patrón</b>	<i>La cancelación de facturas debe poder realizarse con estado de pago e</i>		
	<b>Nombre del parámetro</b>	<b>Nombre de la métrica</b>	<b>Tipo de métrica</b>
	<i>e</i>	<i>Elemento del conjunto de factores</i>	<i>conjunto de factores</i>
		<i>factores</i>	<i>Nominal {totalmente pagado, parcialmente pagado, sin pagar, etc.}</i>
<b>Extensión</b>	<b>Identificador: 9.1 Nombre:</b> Generación de nota de credito		
	<i>La cancelación de una factura pagada o parcialmente pagada debe generar una nota de crédito</i>		
<b>Comentarios</b>	<comentarios>		

<b>Identificador: 10 Nombre:</b> Diferencias entre facturas aprobadas y valoración de existencias	
<b>Módulo</b>	Finanzas
<b>Palabras clave</b>	Cálculo, coste, precio
<b>Versión</b>	1 (marzo 2008)
<b>Autor</b>	GESSI-UPC
<b>Fuente</b>	Libro de requisitos proporcionado por CITI
<b>Objetivo del Patrón</b>	Reflejar diferencias entre las facturas aprobadas y la valoración de existencias
<b>Descripción</b>	El patrón expresa la necesidad de determinar las diferencias entre las facturas aprobadas y la valoración de existencias de forma automática.
<b>Parte fija del patrón</b>	<i>El sistema deberá reflejar las diferencias entre facturas aprobadas y la valoración de existencias</i>
<b>Comentarios</b>	<comentarios>