REVISTA Universidad EAFIT
Vol. 42. No. 141. 2006. pp. 40-59

Reglas de consistencia entre modelos de requisitos de Un-Método

Carlos Mario Zapata J.
Ingeniero Civil, Especialista en Gerencia de Sistemas Informáticos, Magíster en Ingeniería de Sistemas y Candidato a Doctor en Ingeniería con énfasis en Sistemas. Actualmente Profesor Asistente e de la Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia, Sede Medellín. Integrante del Grupo de Investigación UN-INFO.
cmzapata@unal.edu.co

Sandra Milena Villegas S.
Estudiante de Ingeniería de Sistemas de la Universidad Nacional de Colombia. Actualmente Asistente de Investigación del Proyecto "Extensiones en herramientas CASE con énfasis en formalismos y reutilización", que se realiza en convenio con la Universidad Nacional, la Universidad EAFIT y Colciencias. Integrante del Grupo de Investigación UN-INFO de la Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia.
smvilleg@unal.edu.co

Fernando Arango I.
Ingeniero Civil Universidad Nacional de Colombia. Magíster en Water Resources Planning and Management de la Universidad de Colorado State. Doctor en Ingeniería de Programación e Inteligencia Artificial de la Universidad Politécnica de Valencia. Actualmente Profesor Asociado de la Escuela de Sistemas, Facultad de Minas, Universidad Nacional de Colombia, sede Medellín e Integrante del Grupo de Investigación UN-INFO.
farango@unal.edu.co

Recepción: 06 de octubre de 2005 | Aceptación: 21 de enero de 2006

Resumen[1]

El software debe poseer un enfoque ingenieril que permita asegurar su utilidad, lo cual significa que debe satisfacer las necesidades de sus interesados. En la Escuela de Sistemas de la Universidad Nacional de Colombia, se ha creado Un-Método para el desarrollo de software, que toma elementos de otros métodos como Rup y Cdm, e incorpora otros nuevos. Entre los diferentes artefactos incluidos en Un-Método, aún no se realiza un manejo claramente estructurado de la consistencia, por lo cual en este artículo se define un conjunto de reglas que aplican para la consistencia en Un-Método y se muestra su aplicación mediante un caso práctico.

Palabras Clave

Métodos para el desarrollo de Software
Un-Método
Consistencia
Modelamiento

Consistency rules between requirement models of Un-Método

Abstract[1]

Software must have an engineering approach that enables to ensure its usefulness; this means that it must satisfy its stakeholder's needs. In the Systems School of Universidad Nacional de Colombia, Un-Método for software development has been created; Un-Método is a mix of other methods such as Rup and Cdm, and new methods. In this paper we define a set of rules applied to Un-Método consistency, because this method doesn't show clear structured consistency management. Finally, we show the application of rules in a practical case.

Key words

Software development Methods
Un-Método
Consistency
Modeling

Introducción

La "Crisis del Software" (Gibbs, 1994), se ha caracterizado por los incumplimientos reiterativos de los presupuestos y tiempos de entrega en los proyectos de software, y como respuesta a este fenómeno se dio inicio a finales de los años sesenta a la Ingeniería de Software. Un propósito importante en esta disciplina, es el desarrollo de productos que realmente satisfagan las necesidades de los interesados. El grado en que una pieza de software satisface las necesidades de sus interesados, es un factor primario para evaluar su calidad. A este factor se le ha denominado *utilidad* (Grudin, 1991).

Para garantizar la utilidad, es necesario que el desarrollo del software parta de un conocimiento sustancial de las necesidades de los interesados. Los métodos existentes de desarrollo ofrecen tanto artefactos orientados a representar las necesidades de los interesados, como artefactos orientados a representar el software que da cuenta de dichas necesidades.

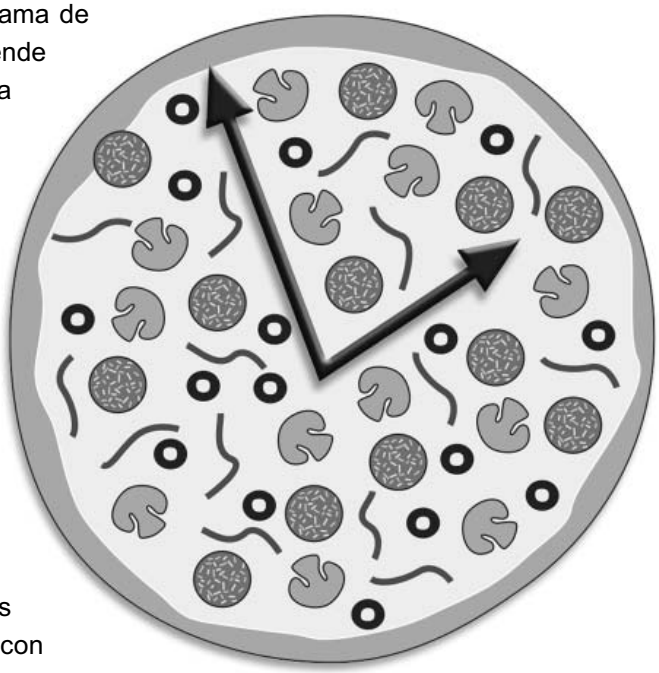
Entre los artefactos que se orientan a representar las necesidades, se destacan modelos de procesos, tales como los tradicionales diagramas de flujo de datos (Gane y Sarson, 1986), diagrama de procesos (Harrington, 1991; Anderson y Wendelken, 1996) y diagrama de actividades (Omg, 2005), además de modelos de objetivos organizacionales tales,

[1] Este artículo se produjo en el marco del proyecto "Extensiones en herramientas CASE con énfasis en Formalismos y Reutilización", cofinanciado por la Universidad Nacional de Colombia, la Universidad EAFIT y Colciencias.

[1] This article was done through the project "Extensions in CASE tools with emphasis in Formalisms and Reuse" (Extensiones en herramientas CASE con énfasis en Formalismos y Reutilización), financed by Universidad Nacional de Colombia, Universidad Eafit, and Colciencias.

como el diagrama de descomposición jerárquica de objetivos incluido en la metodología Kaos (Dardenne et al., 1993). Entre los artefactos que se orientan a representar el software prescrito, vale la pena mencionar los modelos que representan los aspectos estáticos del software, tales como el modelo Entidad - Relación (Chen, 1976) o el diagrama de Clases (Omg, 2005), y los modelos que representan los aspectos dinámicos del software, tales como el modelo de Casos de Uso, el Diagrama de Transición de Estados y el diagrama de Secuencias (Omg, 2005) que muestra la interacción entre los objetos del sistema.

Los modelos de procesos se han orientado a representar las actividades que los interesados llevan a cabo, junto con los datos que necesitan, para luego prescribir un sistema software que provea apoyo automatizado a la realización de las actividades y/o al suministro de los datos requeridos en las mismas. El diagrama de objetivos organizacionales, por su parte, pretende definir el modo como los objetivos asociados a las actividades de los interesados deberían satisfacer los objetivos organizacionales, antes de definir las actividades mismas y prescribir el sistema. En otras palabras, los modelos orientados a procesos se han encaminado a mejorar el sistema actual, mientras que los modelos orientados a objetivos se han encaminado a diseñar el sistema nuevo.



En la Escuela de Sistemas de la Universidad Nacional de Colombia, se ha venido definiendo un método para el desarrollo de software, denominado Un-Método (Arango y Zapata, 2006), que utiliza algunos de los elementos esenciales de los métodos del estado del arte, combinados con elementos de otras disciplinas que permiten representar el problema y su solución, y otros elementos generados en el marco mismo de Un-Método. Son dos objetivos importantes de Un-Método, los siguientes:

- Incorporar de forma coherente las fortalezas de los diferentes métodos de desarrollo del estado del arte. En particular, se han tomado elementos de Cdm (Oracle, 2000), Rup (Jacobson et al., 1999) y Kaos (Dardenne et al., 1993), uniéndolos con artefactos de otras disciplinas, como el modelo causa efecto (Ishikawa, 1986) y formalismos lógicos, y artefactos concebidos en el marco del mismo Un-Método.

- Reglar y minimizar la documentación del desarrollo, definiendo como entregable el conjunto mínimo de modelos requeridos

para asegurar la utilidad del software y otras características de calidad deseables del mismo.

Una característica interesante de Un-Método es su capacidad para articular los modelos orientados a representar los objetivos, con los modelos orientados a representar las actividades que se llevan a cabo en el sistema actual, permitiendo que los modelos de objetivos se usen exitosamente en desarrollos orientados a mejorar el sistema actual. En esta articulación participan modelos como el Diagrama de Objetivos de Kaos, el Diagrama de Procesos de Cdm, el Diagrama Causa-Efecto de Ishikawa, el grafo conceptual de Sowa (Sowa, 1986) y el diagrama de Casos de Uso (Omg, 2005), junto con algunos artefactos propios de Un-Método, tales como la Tabla Explicativa de los Procesos y el método de valoración de la futura pieza de software a partir de los Casos de Uso y el diagrama Causa – Efecto. Debido a que se están usando artefactos que pertenecen a diferentes métodos de desarrollo, se hace indispensable explicitar las reglas necesarias para garantizar que dichos artefactos permanezcan consistentes, con el fin de lograr que la pieza de software que resulte posteriormente, cumpla con la *utilidad* que se mencionó anteriormente.

En este artículo se presentan los criterios que determinan la consistencia entre los artefactos mencionados y que, en consecuencia, determinan la efectividad de Un-Método para el desarrollo de software orientado al mejoramiento de sistemas existentes. Una corriente diferente, que no se analizará en este artículo, tiene que ver con la obtención automática de esquemas conceptuales a partir de especificaciones textuales en lenguaje natural, que constituye una forma diferente de abordar la consistencia a partir de la generación automática (y por ende consistente) de los diferentes diagramas requeridos para el modelamiento de un sistema; para información adicional sobre esta corriente, se puede consultar a Zapata y Arango (2005).

La organización del artículo es la siguiente: en la sección 1 se presenta una descripción de Un-Método, en la que se diferencia de otros métodos de desarrollo de software y se definen algunos de los modelos que se usan en sus fases de definición y análisis; en la sección 2 se justifica el por qué de la necesidad de consistencia en el desarrollo de software; la sección 3 presenta las reglas de consistencia de Un-Método. La aplicación de las reglas mediante un caso de estudio se muestra en la sección 4 y, finalmente, se presentan las conclusiones y trabajos futuros producto del presente estudio.

1. Descripción de Un-Método

La Escuela de Sistemas de la Universidad Nacional de Colombia, viene desarrollando Un-Método, el cual incorpora algunos elementos de otros métodos, tales como los denominados “monumentales” como Custom Development Method - Cdm (Oracle, 2000) y Racional Unified Process - Rup (Jacobson et al., 1999), y los denominados “Ágiles” como Extreme Programming - Xp (Beck, 2000), cuyos objetivos se yuxtaponen; mientras los métodos monumentales buscan la documentación intensiva previa al desarrollo, los métodos ágiles se centran en un desarrollo de prototipos muy rápido, que posibilite la comunicación con los interesados. Si bien Un-Método utiliza elementos de los dos tipos de métodos, se diferencia de ellos en dos aspectos:

- Cdm es un método intensivo en documentación. La gran cantidad de componentes y entregables que hay que llenar para la elaboración de una pieza de software, hace que su ámbito se restrinja notablemente hacia grandes desarrollos; piezas de software de menor tamaño podrían verse entorpecidas con esas grandes cantidades de información requerida. Por el contrario, la mayor cantidad de documentación de Xp se encuentra en el código fuente, pues su enfoque se centra más en la recolección de los requisitos directamente con los interesados y la elaboración rápida de prototipos. En Un-Método, si bien la filosofía de los entregables es similar al Cdm, la cantidad de elementos en los entregables es menor, aunque tratando de garantizar la completitud de los requisitos mediante los modelos que incluye. Además,

incorpora más documentación que Xp, pero sin descuidar la interacción con el interesado, pues se trata de modelar en los diferentes diagramas.

- El punto de partida de Rup son los casos de uso, lo que hace que sea un método muy centrado en la solución. Un-Método, en cambio, se centra en el problema antes de pensar la manera de solucionarlo. Además, por estar basado en Uml y otros artefactos de modelamiento (Kaos, análisis organizacional, etc.), comparte varios de los modelos incluidos en Rup, complementados con otros que contribuyen a identificar y analizar detalladamente los procesos, objetivos y problemas de la organización.

De Un-Método se describen a continuación algunos de los artefactos empleados en la elicitación de requisitos, que es una fase inicial en el proceso de Ingeniería de Software que busca la recolección de las necesidades de los interesados para su procesamiento y comprensión (Christel y Kang, 1992).

1.1 Grafo Conceptual

Este diagrama fue creado por Sowa (1984). A través de este grafo, el analista consigue realizar un "resumen gráfico" de toda la información adquirida a través de su comunicación con el cliente. Un grafo conceptual permite identificar las relaciones entre los diferentes conceptos que hayan aparecido en descripciones previas, referentes al problema.

En un grafo conceptual, los "conceptos" se representan por medio de cajas rectangulares estereotipadas, con el nombre; las relaciones entre los conceptos se representan a través de elipses que en su interior contienen el nombre de la relación y, finalmente, la conectividad entre conceptos y relaciones se realiza a través de líneas. Si bien Sowa emplea las relaciones para ubicar los denominados "roles semánticos", que son los diferentes papeles que puede desempeñar una palabra en relación con el verbo (por ejemplo: agente, locación, tema, experimentador, etc.), en Un-Método se emplean las relaciones para expresar verbos o sintagmas verbales, en tanto que se reservan los conceptos para sustantivos o sintagmas nominales. En la Figura 1 se puede observar un pequeño esquema de un grafo conceptual con la definición genérica de los elementos, como fue presentado por Sowa:

Figura 1. Esquema de un grafo conceptual

Fuente: Sowa, J. F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*. Reading, Addison-Wesley.

1.2 Diagrama de objetivos

Este diagrama está basado en la metodología Kaos (Knowledge Acquisition autOmated Specification) (Dardenne et al., 1993), la cual permite al analista descubrir, a través de la identificación de los objetivos generales de la organización, objetivos más específicos que justifican la construcción del software. El proceso recomendado para la elaboración del diagrama de objetivos en Un-Método se describe a continuación:

En primer lugar se debe tener presente que el diagrama a elaborar debe ser una descomposición progresiva de los objetivos del área de la organización en la que se ubica el problema; para conseguir esto, se deben presentar los objetivos secundarios que son subrogados de los objetivos generales y, a su vez, para estos se deben presentar los objetivos más elementales que los subrogan. Este proceso se debe repetir hasta llegar a los objetivos que se consideren elementales o atómicos. Además de esto, se deben usar relaciones And y Or entre los objetivos (a través de las relaciones And se muestra que todos los objetivos que componen esta relación se deben satisfacer, para que se cumpla el objetivo que subrogan; a través de las relaciones Or, se puede mostrar que si se cumple cualquiera de los objetivos que la componen es suficiente para satisfacer el objetivo que subrogan). En la Figura 2 se muestra un esquema de la forma como se estructura un diagrama de objetivos; en ella se muestran relaciones And y Or, resaltando los niveles

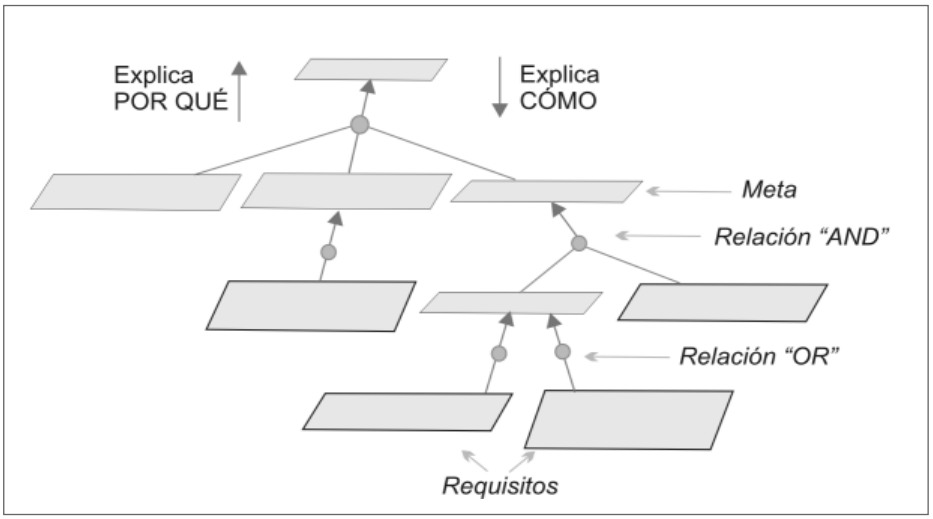
que corresponderían a “metas” y a “requisitos”. Además, se muestra cómo, dependiendo de la orientación en que se lea el diagrama, se puede apreciar el por qué (si la lectura se realiza de abajo hacia arriba) de la necesidad de cumplir con un objetivo, y el cómo (si la lectura se realiza de arriba hacia abajo) se alcanzaría un objetivo.

1.3 Modelo de procesos

El modelo de procesos es un diagrama que muestra las actividades de la organización y la forma como éstas se llevan a cabo. A través de este modelo se puede ver cómo la organización satisface sus objetivos; dicho modelo proviene del análisis organizacional, y de él Harrington presenta algunas versiones preliminares (Harrington, 1991), pero además es usado en herramientas actuales como el Oracle ® Designer (Anderson y Wendelken, 1996).

El modelo de procesos muestra el flujo de información; es decir, la manera como se producen entradas y salidas de datos entre un proceso y otro. En un modelo de procesos se incluyen los siguientes elementos:

Figura 2. Esquema de un diagrama de objetivos



Fuente: Construcción de los autores, a partir de Dardenne *et. al.* (1993).

Actor: puede ser una persona, dependencia o grupo de personas que realiza algún tipo de acción, involucrada dentro del conjunto de pasos para satisfacer los objetivos de la organización. Dentro de la simbología del modelo, por cada actor se debe trazar un carril horizontal, en el que se especificarán las acciones de las que es responsable dicho actor.

Proceso – Acción: es una actividad que puede ser considerada como básica dentro de la organización y se representa a través de un rectángulo rotulado con su respectivo nombre. Un proceso es considerado una actividad de trabajo discreta, pues tiene principio y fin (representados a través de los dos lados opuestos del rectángulo).

Evento: define los hechos externos o condiciones internas al sistema (distintas al inicio o finalización de un proceso interno), que determinan el inicio de un conjunto de procesos asociados por relaciones de precedencia. Un evento se representa por una flecha gruesa, rotulada con el nombre del evento. Si un evento causa el inicio de uno o más procesos, se denomina *disparador*; si el evento es el efecto de la culminación de uno o más procesos se denomina *resultado*.

Condiciones: corresponden a verificaciones de hechos particulares necesarias dentro de un proceso, ya que, dependiendo de si se cumplen o no, se debe seguir una secuencia de pasos específica. Las condiciones se representan a través de rombos, de cuyas puntas deben salir flujos rotulados con los nombres de las alternativas de cada condición (por ejemplo “si” o “no”).

Datos – Almacenamientos: representan información generada o requerida en los procesos. Se simbolizan a través de íconos rotulados con el nombre de los datos. Para una mejor descripción de la información, se debe elaborar una tabla denominada *Diccionario de datos* (Tabla 1).

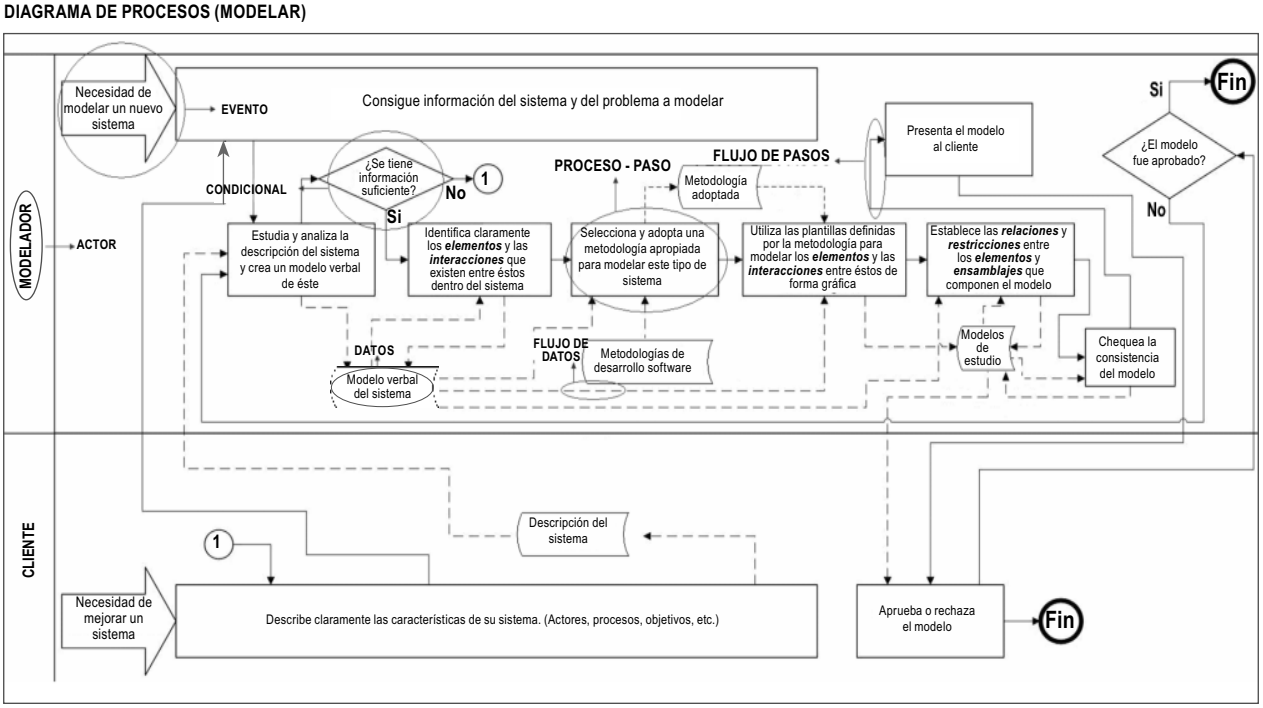
Tabla 1. Formato del Diccionario de datos

Nombre	Acrónimo	Componentes
<Nombre del dato>	*<Rótulo con el que hará referencia al dato posteriormente>.*	*<Nombre de los datos componentes del dato>.*

Flujos: también llamados *intercambios*, pueden ser de datos o de cosas físicas, y los flujos temporales que muestran una secuencia de control donde haya necesidad de un orden particular entre los pasos. En el diagrama de procesos, los flujos se representan por medio de flechas unidireccionales, que suelen ser continuas si se trata de flujos que indican secuencia entre los procesos y punteadas si se trata de flujos de datos.

Una mayor ilustración de la forma como se estructura un diagrama de procesos, se puede observar la Figura 3.

Figura 3. Ejemplo de un diagrama de procesos con sus componentes



1.4 Tabla explicativa de los procesos

Este artefacto se ha propuesto para Un-Método, con el fin de clarificar la información concerniente a los diferentes procesos de la organización. Además, sirve como vínculo entre los diagramas de procesos, objetivos y causa – efecto, siguiendo las reglas de consistencia que se establecen en la sección siguiente. Las componentes de la tabla explicativa de los procesos se pueden apreciar en la Tabla 2.

Tabla 2. Formato de una tabla explicativa de procesos

Nombre	Objetivo	Duración / Frecuencia	Cómo / Dónde	Problemas	Reglas
<Nombre con el que se hará referencia al proceso>	<QUÉ lleva a cabo el proceso, Razón de ser del proceso. NOTA: No se debe repetir la descripción de la información requerida o producida por el proceso>	<CUÁNTO TIEMPO/ CUÁNTAS VECES: - Duraciones actual, deseable y máxima del proceso. - Número de veces que se lleva a cabo.>	< Resumen de la forma como se lleva a cabo el proceso (enfoque). Lugar donde se lleva cabo NOTA: No se debe repetir una descomposición del proceso en sus subprocesos>	<Dificultades actuales en la ejecución del proceso, en cuanto a su capacidad de lograr los resultados esperados con la calidad deseada (incluido el costo) y en el tiempo esperado>	<Reglas del negocio: - Restricciones que se deben tener en cuenta al llevar a cabo el proceso. - Fórmulas con que se llevan a cabo los cálculos. NOTA: Se deben colocar referencias a apéndices, si es necesario>

1.5 Diagrama causa-efecto

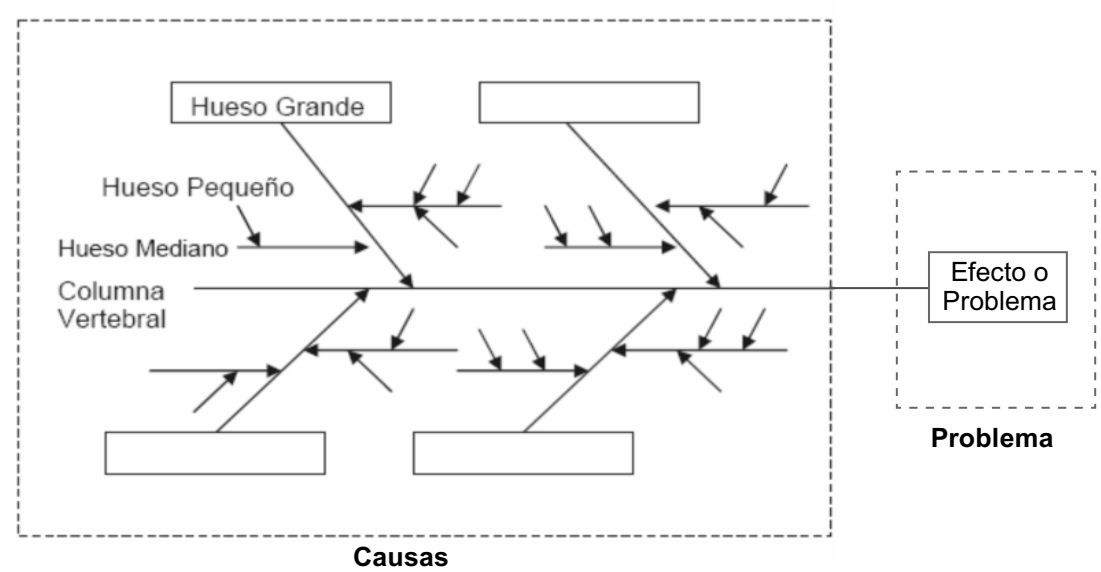
Este diagrama permite al analista estructurar y jerarquizar los problemas que identifica en el discurso proporcionado por el cliente para, de esta forma, tomar decisiones respecto de cuál deberá ser el área en la que se enfoca su trabajo. En otras palabras, a través del análisis que aquí se origina, el analista podrá decidir qué problemas deberá atacar en su totalidad y cuáles podrá omitir o atacar parcialmente; no debe alejarse del objetivo general que justifica la creación del software.

Este diagrama también es conocido como *espina de pescado*, por la similitud de su apariencia física con la de un esqueleto de pez, o como Diagrama de Ishikawa en honor a su creador (Ishikawa, 1986).

Para la elaboración del diagrama Causa-Efecto, se puede proceder de dos formas: la primera de ellas consiste en listar todos los problemas identificados (tipo “lluvia de ideas”), para luego intentar jerarquizarlos y estructurarlos identificando cuáles son principales y cuáles son sus causas, realizando reiteradas veces este procedimiento hasta que se logre recorrer todos los problemas identificados, o hasta que las causas que se tengan sean consideradas atómicas. La segunda forma de elaborar este diagrama consiste en identificar los problemas principales y ubicarlos como “Huesos primarios” y, posteriormente, comenzar a identificar causas secundarias, que se ubicarán en “Huesos pequeños”, que se desprenderán todos de las ramas principales. Como es de esperarse, en un diagrama Causa – Efecto, se debe identificar un problema principal que logre encerrar la problemática del área en la que se concentrará el trabajo del analista, por lo cual es recomendable prestar especial cuidado a este aspecto, ya que de la correcta identificación de dicho problema depende si se obtiene o no un buen y completo diagrama.

Para una mayor ilustración de la forma como se estructura el diagrama Causa – Efecto, se puede observar la Figura 4.

Figura 4. Esquema de un diagrama Causa – Efecto



Fuente: Adaptación de Ishikawa (1986).

1.6 Diagrama de casos de uso

Los diagramas de casos de uso son diagramas de Uml para modelar aspectos funcionales del sistema. Estos diagramas permiten reconocer claramente los límites del sistema y las relaciones con su entorno (Omg, 2005). Para una mejor descripción, consultar a Fowler (2003) o la especificación de Uml 2.0, presentado por el Omg (2005).

Los diagramas de casos de uso comúnmente contienen los actores, que representan los roles que un usuario puede desempeñar en el sistema, y los casos de uso, que son operaciones o tareas específicas que se realizan tras una orden de algún agente externo, sea desde la petición de algún actor o desde la invocación de otro caso de uso.

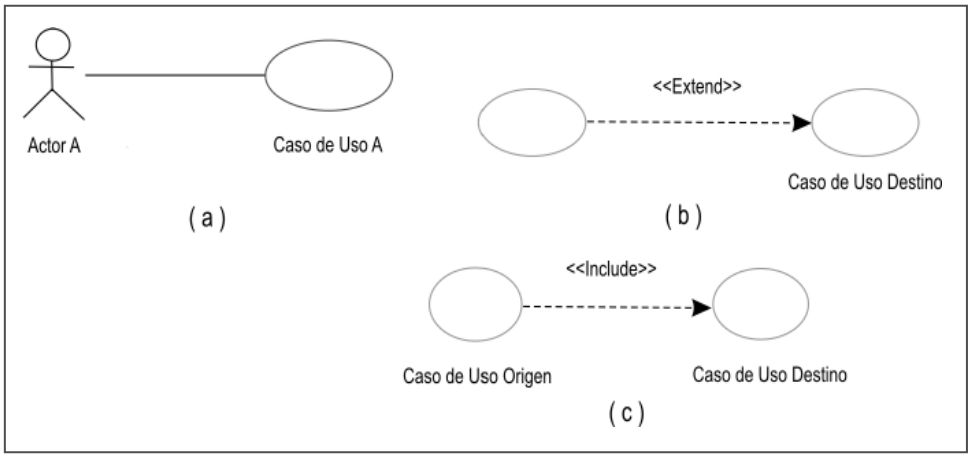
Entre los casos de uso se pueden establecer relaciones de inclusión (indica que el caso de uso base u origen incluye el comportamiento descrito por el caso de uso destino) y extensión (muestra que el caso de origen extiende o amplía el comportamiento del caso de uso destino).

En la Figura 5 se muestra la representación gráfica de un caso de uso; en ella se observa un ícono que simboliza el rol del actor que se relaciona con el caso de uso. En cuanto a estos, se simbolizan a través de óvalos en los que se inscribe el nombre del caso de uso y se muestran las respectivas flechas que muestran las relaciones entre ellos.

2. La necesidad de consistencia en el desarrollo de software

Cada pieza de software se modela mediante un conjunto de artefactos, y es ésta la razón por la cual se hace necesario articularlos, de tal modo que los resultados sean consistentes; cuando ello no se hace, el analista tiene que concentrarse en corregir errores que tienen su origen en las fases iniciales del desarrollo del software y pierde gran parte del esfuerzo necesario para realizar otras tareas. Zowghi y Gervasi enuncian otras razones sobre la necesidad de consistencia, tales como: la evolución de los requisitos de los interesados, que hace que los diferentes artefactos referentes a la conceptualización de una solución, deban permanecer correctos para poder soportar los múltiples cambios que plantean los interesados, y el hecho de que los requisitos pueden ser levantados por diferentes personas que introducen sus propias apreciaciones en el proceso de desarrollo del software, generando diferentes puntos de vista del mismo problema (Zowghi y Gervasi, 2002).

Figura 5. En a) representación gráfica de un caso de uso, en b) representación de extensión entre dos casos de uso, y en c) representación de inclusión entre dos casos de uso



Fuente: Los autores, a partir de Omg (2005).

Muchos de los errores a los que se ve enfrentado un grupo de desarrollo de software, son producto de las inconsistencias entre los diferentes artefactos; esto se debe a que cada artefacto modela una porción específica del dominio con sus propias características, siendo muy probable que se modelen elementos comunes, y es ahí donde surge la necesidad de consistencia.

Cuando se aplica un método específico de desarrollo de software, es posible que cada elemento del dominio se pueda modelar de diferente manera en los diferentes artefactos que propone el método, pero debe ser claro que se trata del mismo elemento, para que a la hora de realizar la implementación, el desarrollador no se encuentre con ambigüedades o redundancia en la información, evitándose así grandes molestias en la fase de mantenimiento, que es quizás la tarea más ardua de todo el proceso.

La necesidad de consistencia entre los diferentes modelos que representan una solución, justifican la necesidad de contar con un conjunto de reglas de consistencia; en particular, esa necesidad de contar con un inventario de las reglas correspondientes a los artefactos se hace manifiesta en Un-Método, lo que permitirá a los analistas que aplican este método obtener mejores resultados en el desarrollo de sus productos. Es importante aclarar que hasta el momento se ha trabajado consistencia en Uml con diferentes métodos (Omg, 2005), pero aún no se ha trabajado con diagramas diferentes a Uml. Es esta la razón que más ha motivado esta propuesta.

3. Reglas de consistencia de Un-Método

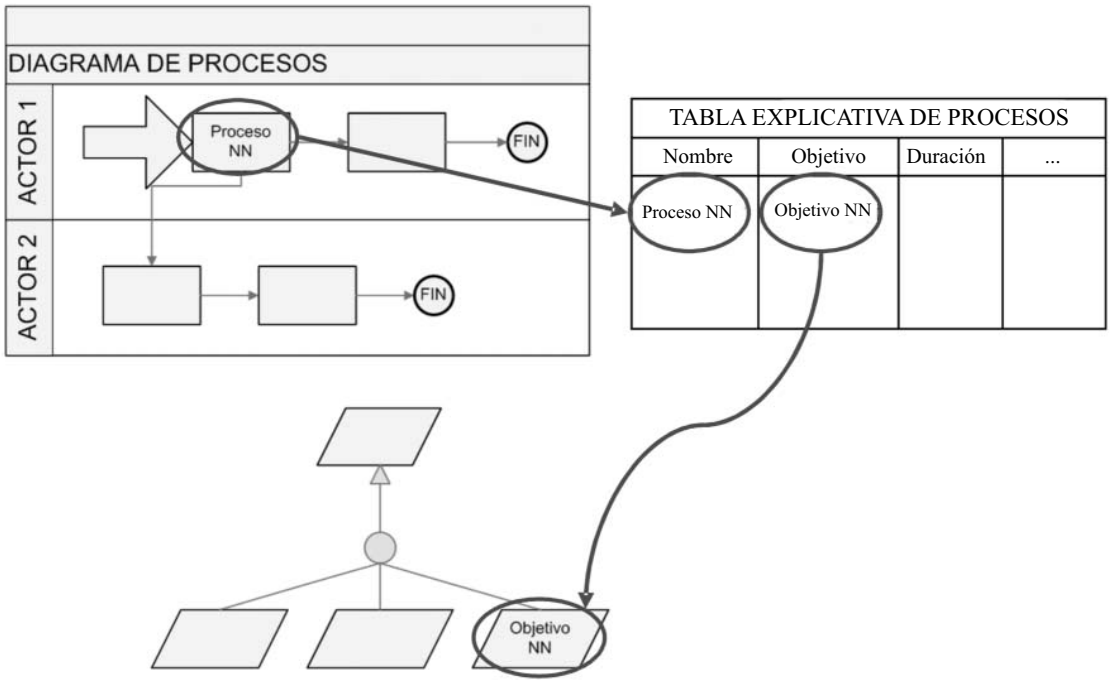
En esta sección se define un conjunto de reglas que permiten la realización de chequeos de consistencia entre los diferentes artefactos de Un-Método.

3.1 Regla 1

Los objetivos que se enuncien en la tabla explicativa del diagrama de procesos, deben estar contenidos en el diagrama de objetivos.

Observaciones: En la Figura 6 se puede observar un esquema de la regla. Se debe tener en cuenta que los procesos de la tabla mencionada, por razones de consistencia, deben estar consignados en el diagrama de procesos. Esto permite establecer el vínculo entre los procesos u operaciones del dominio y los grandes objetivos de la organización.

Figura 6. Consistencia entre diagrama de procesos y diagrama de objetivos

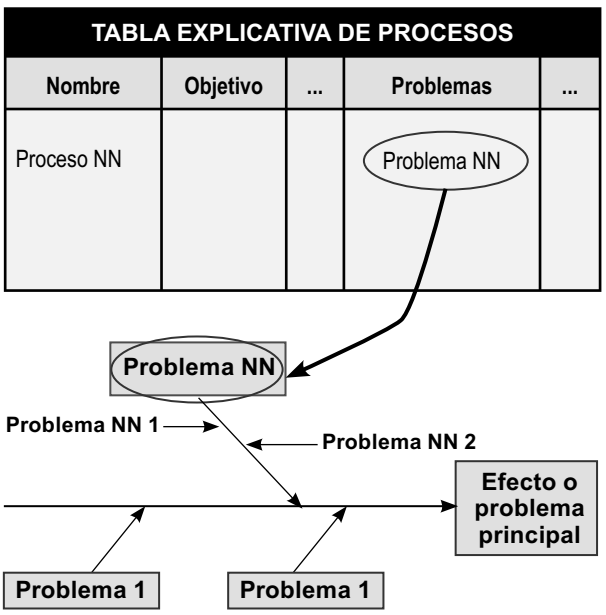


3.2 Regla 2

Los problemas que se identifican a través del desarrollo del diagrama de procesos, deben estar consignados en el diagrama Causa – Efecto.

Observaciones: A través de este método se justifica la construcción del software (tiene sentido mostrar que la forma como se desarrollan los procesos presenta los problemas para los cuales se debe buscar solución). Para cumplir con esta regla, es necesario tomar en cuenta que en la columna etiquetada con el nombre de "Problemas" de la tabla explicativa de procesos, se deben consignar problemas que puedan ser identificados en el diagrama Causa – Efecto, como se muestra en la Figura 7.

Figura 7. Consistencia entre diagrama Causa-Efecto y diagrama de procesos



Fuente: Los autores.

3.3 Regla 3

Los objetos a los que se refieren los casos de uso deben aparecer explícitamente como "conceptos" en el grafo conceptual.

Observaciones: Como se mencionó anteriormente, en el grafo conceptual están consignados todos los "conceptos" que el analista logra extraer del discurso proporcionado por el cliente; los casos de uso son acciones que se materializan sobre los diferentes objetos del discurso y por ello deberían estar incluidos en el grafo conceptual. Por ejemplo, para el caso de uso "Registrar Producto", debería existir un concepto denominado "Producto" en el grafo conceptual.

3.4 Regla 4

El problema principal del diagrama Causa – Efecto debe corresponder a la falta de satisfacción en un objetivo de la organización.

Observaciones: Los problemas que motivan el desarrollo de una pieza de software, se suelen detectar cuando algunos de los objetivos de la organización no se alcanzan o se satisfacen a medias. Eso sugiere una relación directa entre la cabeza del diagrama Causa – Efecto y el diagrama de Objetivos.

3.5 Regla 5

Los casos de uso de la solución informática propuesta, deben atender en gran medida las causas que ocasionan el problema principal identificado en el diagrama Causa – Efecto.

Observaciones: Esta regla es de vital importancia porque justifica en gran medida la creación del software, puesto que el diagrama de casos de uso muestra una aproximación clara a lo que a futuro serán las interfaces; podría decirse que a partir de este diagrama es fácilmente identificable la solución que se está planteando y, si dicha solución no se valida a través de la búsqueda de un valor que permita brindar una aproximación al porcentaje del problema (desmembrado en el diagrama causa efecto) que será atendido, entonces no tendrá sentido crear una pieza de software cuyas funciones en nada contribuyan al mejoramiento de la forma como se llevan a cabo los procesos (aquí se identifican los problemas consignados en el diagrama causa efecto).

Zapata y Arango proponen un método que permite vincular los problemas identificados en el diagrama Causa – Efecto, con las interacciones de los casos de uso de la solución informática. En la Tabla3 se muestra la forma como deben describirse las causas anotadas en el diagrama Causa - Efecto (Zapata y Arango, 2004).

Luego de completar la Tabla 3, se debe realizar una tabla de trazabilidad (véase la Tabla 4), que permite identificar los vínculos entre las funciones del diagrama de casos de uso y las causas del diagrama Causa – Efecto que son atendidas.

Tabla 3. Formato para la descripción de las causas del diagrama Causa - Efecto. Las categorías presentes en la tabla son: C: Carencia, MC: Mala calidad y FO: fallas en la oportunidad; todas ellas se refieren a la información asociada con la causa

Nº	Nombre	Descripción	Categoría			
			C	MC	FO	Pi,j
Identificación de la causa (por ejemplo, $C_{1,1}$)	Nombre correspondiente al problema (causa) en el diagrama Causa – Efecto	Descripión del problema / causa				

Fuente: Zapata, C. M. y Arango, F. (2004). "Alineación entre Metas Organizacionales y Elicitación de Requisitos del Software". En: *Revista Dyna*. No. 143. pp. 101-110.

En la tabla 4 aparecen las siguientes columnas: nombre de la función del caso de uso, causas atendidas (es decir, problemas que la función ataca total o parcialmente), $P_{i,j}$ (porcentaje de relevancia de la causa $C_{i,j}$ en el subproblema Sp_i), Q_i (porcentaje de relevancia del subproblema Sp_i en el problema principal PP), $A_{i,j}$ (coeficiente de cumplimiento, cuyo valor oscila entre 0 -si la función no atiende la causa definida- y 1 -si la función atiende completamente la causa definida-) y, finalmente, %Acum (que corresponde al porcentaje de atención del caso de uso sobre el problema principal PP). El porcentaje de atención del caso de uso %Acum se calcula con base en la siguiente fórmula:

$$\%Acum = \sum_{k=1}^{\tilde{n}} \left(\sum_{i=1}^{n} Q_i * \left(\sum_{j=1}^{m} P_{i,j} * A_{i,j} \right) \right)$$

Donde *ñ* es el número máximo de funciones del caso de uso, *m* la cantidad máxima de causas asociadas a cualquiera de los subproblemas y *n* la cantidad de subproblemas que se asocian al problema principal. Como una última observación a esta regla, cabe anotar que la idea del método propuesto es obtener un valor considerablemente alto para el %Acum, lo que indicará que con la solución informática propuesta (cuyas funciones se describen en los diagramas de casos de uso) se atiende gran cantidad de las causas que ocasionan el problema principal.

Tabla 4. Tabla de trazabilidad que sirve de vínculo entre las funciones del diagrama de casos de uso y las causas del diagrama Causa – Efecto que son atendidas

Nº	Función del diagrama de casos de Uso	Causas atendidas	Pi,j	Qi,j	Ai,j	%ACUk
F1						
F1						
FN						
	Total					

Fuente: Zapata, C. M. y Arango, F. (2004). "Alineación entre metas organizacionales y elicitación de requisitos del software". En: *Revista Dyna.* No. 143. pp. 101-110.

4. Caso de estudio para el chequeo de las reglas

En este apartado se presentan segmentos de algunos de los diagramas que modelan el sistema de una pizzería, cuyo problema a resolver es la entrega de pedidos, para ejemplificar las reglas de consistencia entre los diferentes artefactos. Se inicia con el "modelo verbal", que es una expresión verbalizada de las características del dominio (Zapata y Arango, 2005) y en la cual se reflejan las necesidades del interesado.

Modelo verbal

La pizzería lleva ya cerca de un año de actividad y hasta ahora no se ha conseguido ganar tanto dinero como se había esperado. Esta pizzería cuenta con un despachador, un cocinero y dos repartidores, y ofrece a los clientes diversos tipos y tamaños de pizza, además de la posibilidad de ordenar aditivos. La pizzería tiene una zona de cobertura determinada, y con el fin de competir frente a otras en la zona, se ha estado ofreciendo al cliente una promoción, que consiste en entregarle el pedido totalmente gratis si tarda en ser entregado más de 30 minutos. Esta promoción ha atraído buena cantidad de clientes, pero también se ha convertido en una de las principales fuentes de pérdida de dinero, puesto que muy a menudo los

repartidores no consiguen llevar todas las pizzas a tiempo. A inconformidad de los repartidores, se ha impuesto en el reglamento que cada pedido entregado tarde, deberá ser pagado por el repartidor responsable, con el fin de evitar tanta pérdida de dinero y conseguir que los repartidores se esfuercen más en su trabajo.

En la Figura 8 se muestra el diagrama de procesos de la situación actual de la pizzería, que contiene los flujos y procesos involucrados en la solicitud y posterior entrega de un pedido. El enlace de este diagrama con los demás se hace por medio de la tabla explicativa de procesos que se muestra más adelante.

En la Tabla 5, (por efectos de ilustración de este ejemplo se eliminaron algunas columnas; se recuerda al lector que en un proceso completo de modelamiento se deben incluir todas las columnas propuestas en la Tabla 2), se presenta la Tabla explicativa para algunos de los procesos del diagrama de la Figura 8. En dicha tabla se puede observar la correspondencia con el diagrama Causa – Efecto y con el de objetivos mostrados en las Figuras 9 y 10 respectivamente; obsérvese además que dicha correspondencia es la enunciada en las reglas 1 y 2. También se observa que el problema principal del diagrama Causa – Efecto, corresponde a la insatisfacción del objetivo principal del Diagrama de Objetivos, como lo enuncia la Regla 4; cabe aclarar que el objetivo insatisfecho podría no ser el principal, sino un objetivo ubicado en una rama intermedia.

Figura 8. Diagrama de procesos situación actual – Sistema de entrega de pedidos de una pizzería

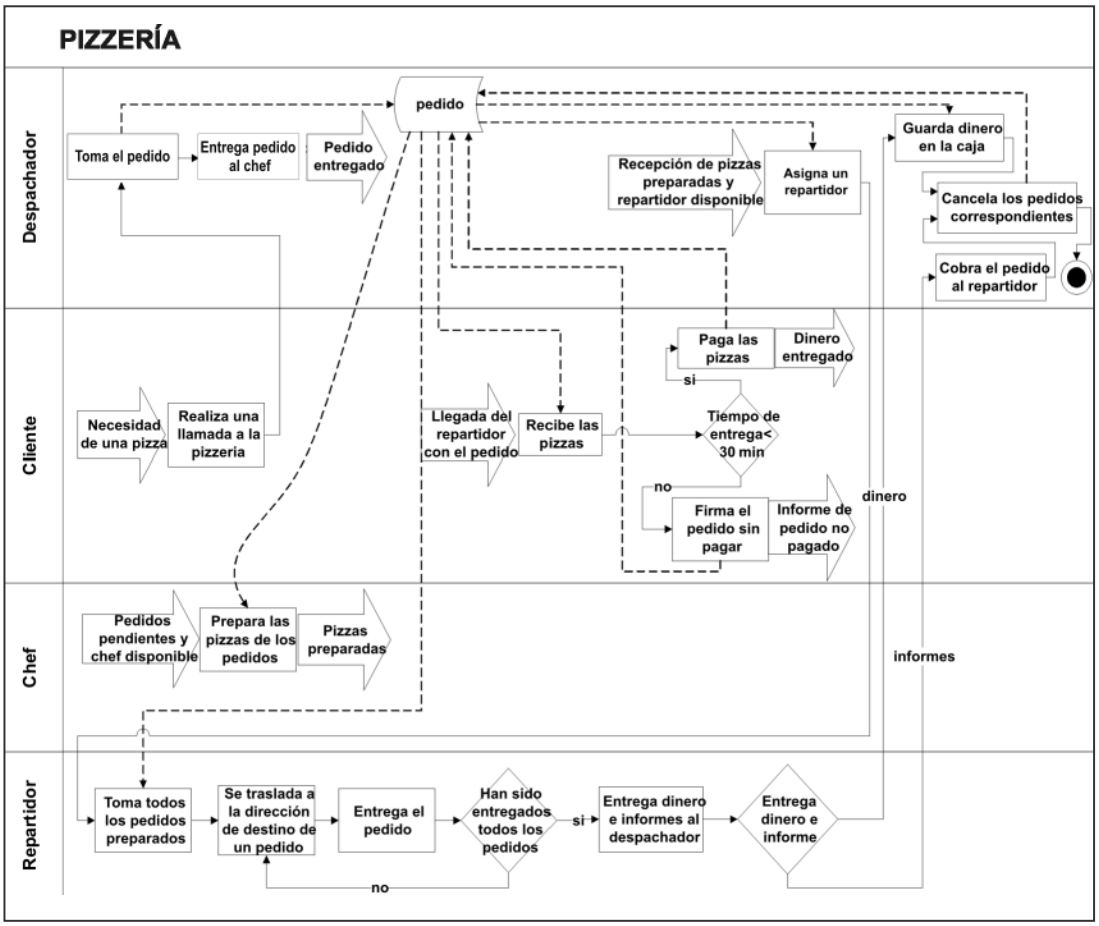


Tabla 5. Tabla explicativa de algunos procesos del diagrama mostrado en la Figura 11

Nombre	Objetivo	Responsable	Datos de Entrada / Salida	Problemas
Entrega el pedido	Satisfacer las necesidades del cliente en relación con los productos de la pizzería	Repartidor	• Productos despachados, pedidos • Productos entregados	• El producto entregado no corresponde al producto solicitado por el cliente
Asigna un repartidor	Garantizar el envío del producto al cliente	Despachador	• Productos elaborados, repartidores disponibles, pedidos • Productos asignados, pedidos	• No hay repartidores disponibles
Toma los pedidos	Garantizar el envío del producto al cliente	Repartidor	• Productos asignados, pedidos • Productos despachados, pedidos	• El repartidor toma demasiados pedidos
Prepara las pizzas de los pedidos	Satisfacer las necesidades del cliente en relación con los productos de la pizzería	Chef	• Pedidos • Productos elaborados	• No hay suficientes ingredientes para elaborar todos los productos solicitados • La cantidad de productos solicitados excede las capacidades de producción del chef • Exceso de llamadas y pedidos
Se traslada a la dirección de destino de un pedido	Garantizar el envío del producto al cliente	Repartidor	• Pedido, dirección de destino • Destino alcanzado	• La dirección está errónea • La dirección está fuera de la zona de cobertura • Los destinos de los pedidos se encuentran muy distantes unos de otros

Figura 9. Diagrama Causa – Efecto de entrega de pedidos de una pizzería

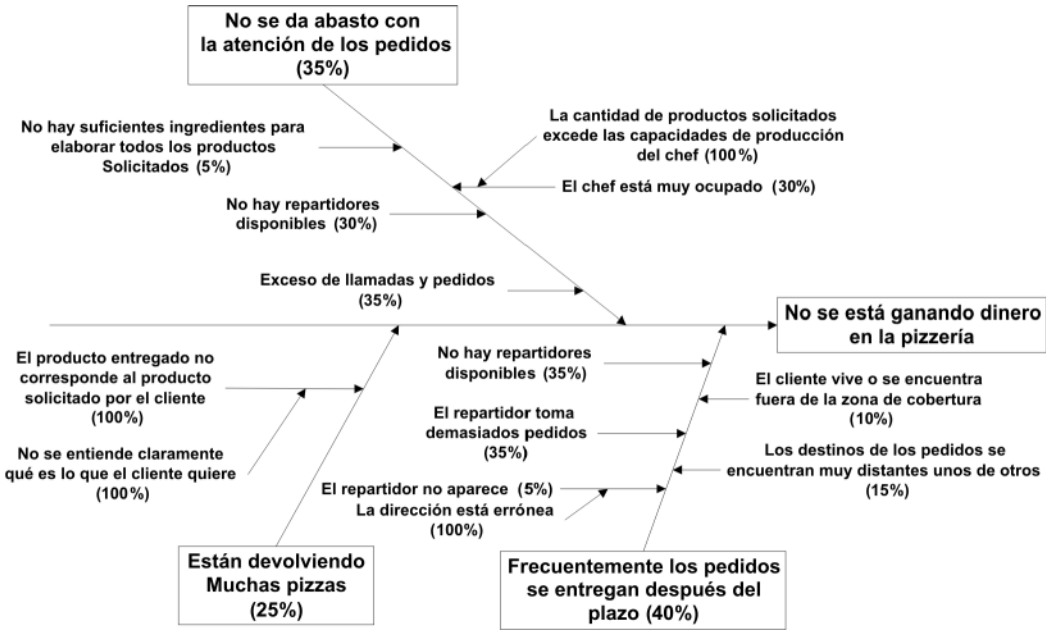


Figura 10. Diagrama de objetivos – Sistema de entrega de pedidos de una pizzería

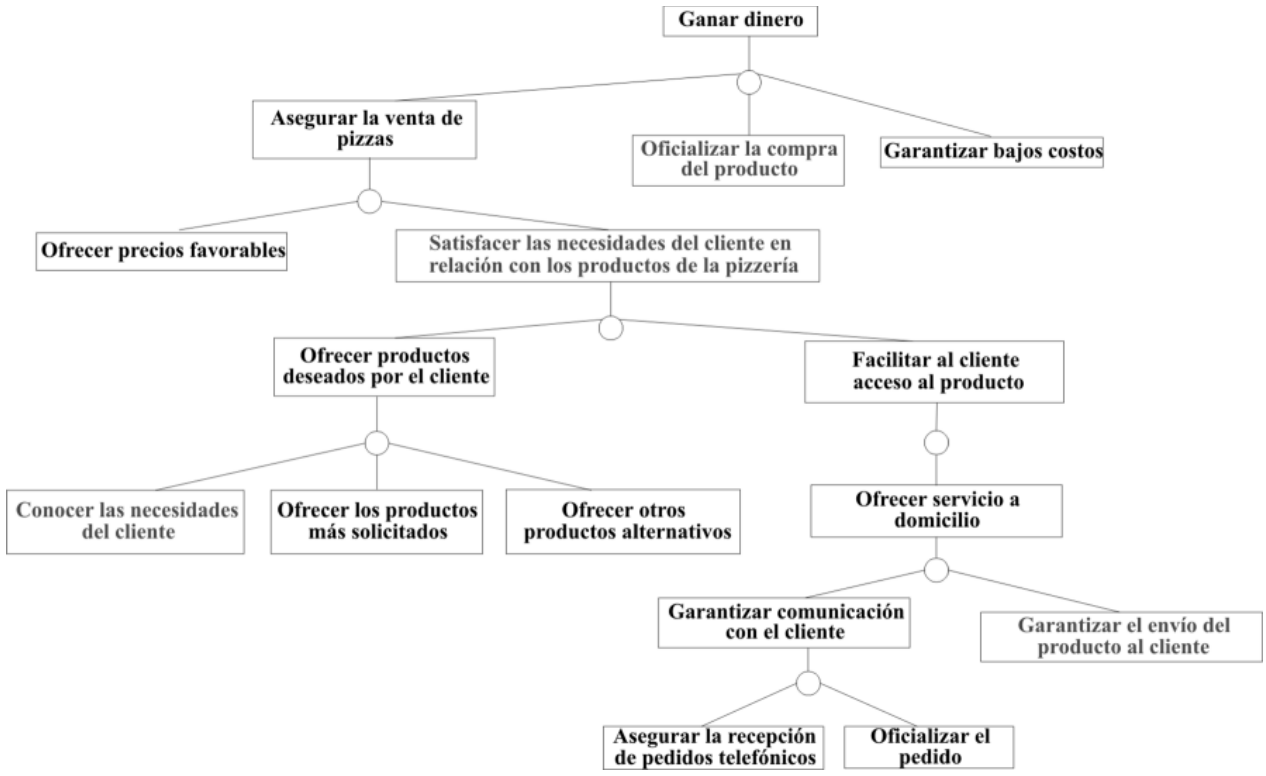


Figura 11. Porción del grafo conceptual – Sistema de entrega de pedidos de una pizzería

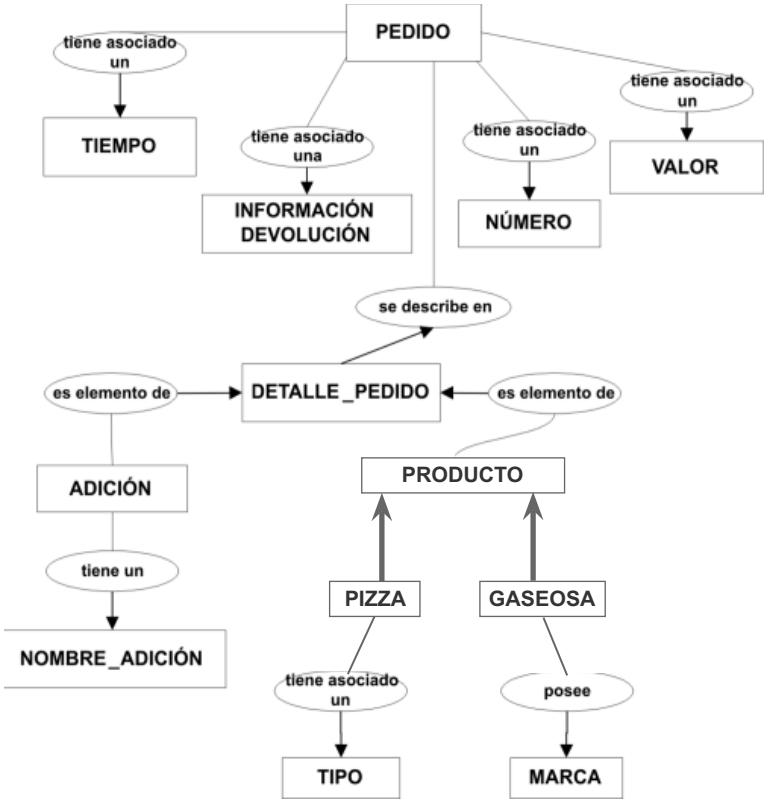
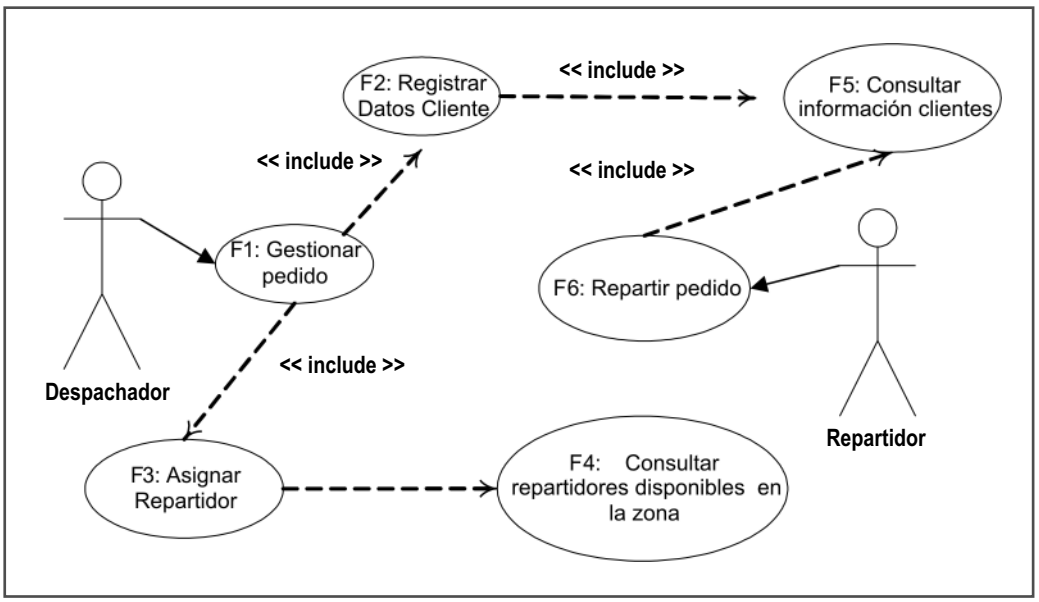


Figura 12. Ejemplo de un caso de uso –Sistema de entrega de pedidos de una pizzería



Para ilustrar la idea propuesta en la Regla 5, a continuación se aplica el método para el ejemplo de la pizzería. Como ya se mencionó, en primer lugar se realiza la descripción de las causas (todas ellas contenidas en el diagrama Causa – Efecto) y posteriormente la tabla de trazabilidad entre las causas y los casos de uso. Las Tablas 6 y 7 muestran los resultados.

Según los resultados obtenidos, los casos de uso propuestos contribuirán a la solución de un 77% de los problemas identificados en el análisis, con lo cual se evidencia que existe consistencia "implícita" entre el diagrama Causa – Efecto y el diagrama de Casos de Uso, que está fundamentada en un análisis teórico.

Tabla 6. Descripción de las causas anotadas en el diagrama de la Figura 12. Recuérdense las convenciones: C – carencia, MC – Mala Calidad y FO falta de oportunidad

Nº	Nombre	Descripción	Categoría			
			C	MC	FO	Pi,j
$C_{1,1}$	No hay repartidores disponibles	Las utilidades de la empresa impiden contratar más repartidores, razón por la cual los pocos repartidores con los que se cuenta permanecen ocupados	X			35
$C_{1,2}$	El cliente vive o se encuentra fuera de la zona de cobertura	En ocasiones la empresa, buscando conseguir más clientes, se compromete a entregar pedidos en sitios ubicados fuera de la zona de cobertura			X	10
$C_{1,3}$	El repartidor toma demasiados pedidos	La carencia de repartidores induce a que estos tengan que comprometerse a entregar muchos más pedidos de los que sus capacidades reales permiten			X	35
$C_{1,4}$	El repartidor no aparece	En ocasiones se presentan confusiones con la información asociada a los clientes; esto propicia que el repartidor pueda "embolatarse" buscando direcciones inadecuadas			X	5
$C_{1,5}$	Los destinos de los pedidos se encuentran muy distantes unos de otros	La no distribución por zonas a los repartidores, ocasiona que un mismo repartidor tenga que recorrer grandes distancias para entregar los diferentes pedidos			X	15
$C_{2,1}$	Exceso de llamadas y pedidos	La promoción ofrecida por la pizzería atrajo demasiada cantidad de clientes, lo que ocasionó una "avalancha" de pedidos que en gran número de casos no son llevados a tiempo			X	35
$C_{2,2}$	No hay suficientes ingredientes para elaborar todos los productos solicitados	La cantidad de pedidos a la que se está viendo enfrentada la pizzería, hace incalculable la reserva de ingredientes que el chef debe mantener para satisfacer la demanda	X			5
$C_{2,3}$	No hay repartidores disponibles	Las utilidades de la empresa impiden contratar más repartidores, razón por la cual los pocos repartidores con los que se cuenta permanecen ocupados	X			30
$C_{2,4}$	El chef está muy ocupado	La gran cantidad de pedidos hace que el chef esté todo el tiempo ocupado, lo cual ha propiciado disminución en su rendimiento	X			30
$C_{3,1}$	El producto entregado no corresponde al producto solicitado por el cliente	Algunas veces, debido a la cantidad de pedidos, los empleados de la pizzería han enfrentado confusiones que finalizan con la entrega de un producto que no es el solicitado		X		100

Tabla 7. Trazabilidad entre las causas anotadas en el diagrama de la Figura 12 y los casos de uso que se muestran en la Figura 16

No.	Función del diagrama de casos de Uso	Causas atendidas	Pi,j	Qi,j	Ai,j	%ACUk
F1	Gestionar pedido	$C_{1,3}$; $C_{2,1}$; $C_{2,4}$	0.35; 0.05; 0.30	0.40; 0.35; 0.35	0.7; 0.4; 0.4	0.147
F2	Registrar Datos Cliente	$C_{2,1}$; $C_{2,4}$	0.35; 0.30	0.35; 0.35	0.2; 0.1	0.035
F3	Asignar repartidor	$C_{1,1}$; $C_{1,3}$	0.35; 0.35	0.40; 0.40	0.4; 0.2	0.084
F4	Consultarrepartidores disponibles en la zona	$C_{1,5}$	0.15	0.40	1	0.06
F5	Consultar información de los clientes	$C_{3,1}$	1	0.25	0.8	0.2
F6	Repartir pedido	$C_{2,1}$	0.35	0.35	0.2	0.245
	Total					0.771

Conclusiones y trabajos futuros

La Consistencia constituye un nuevo campo de estudio que cobra mayor importancia en el ámbito de la ingeniería de software, dado que los métodos de desarrollo de software se componen de artefactos que pueden ser desarrollados por diferentes personas (generando puntos de vista diferentes) y por la evolución de los requisitos del interesado. La consistencia debe ser un aspecto fundamental a tener en cuenta en cualquier método de desarrollo de software, porque de ella depende en gran medida la calidad de los resultados que se obtienen.

En este artículo se han descrito los diferentes modelos que hacen parte de las fases de definición y análisis de Un-Método, explicando a su vez las diferencias de este método con algunos otros de la literatura especializada, como Rup, Xp o Cdm. Igualmente, se han propuesto y ejemplificado algunas reglas de consistencia que se deben tomar en consideración para los siguientes modelos de Un-Método: grafo conceptual, diagrama de procesos, tabla explicativa de los procesos, diagrama de objetivos, diagrama causa – efecto, diagrama de clases y diagrama de casos de uso. Estas reglas de consistencia suministran un derrotero para los analistas que usen Un-Método como estrategia de desarrollo de software, con el fin de determinar la validez de los diferentes diagramas que se realicen dentro de las fases de definición y análisis de un problema particular.

En el proceso de modelamiento que se realiza con Un-Método, la consistencia, entre otras cosas, permite al analista enlazar la información de tal forma que, partiendo de la definición de unas metas organizacionales, se llega a la solución del problema; esto justifica en gran medida la construcción del software, facilitando incluso al cliente mayor comprensión del problema al que está enfrentado.

Como posibles trabajos futuros que se pueden desprender de este artículo, se mencionan los siguientes:

- La búsqueda e incorporación en Un-Método de nuevas reglas de consistencia entre los artefactos que incluye

- La programación de estas reglas utilizando una herramienta MetaCASE, que permita la ampliación de los modelos para cubrir las necesidades correspondientes a la consistencia intermodelos
- La utilización de las reglas de consistencia definidas en el proceso de generación automática de algunos modelos de Un-Método en términos de otros, como una especie de refinamiento realizado vía las reglas de consistencia.

Bibliografía

Anderson, C. y Wendelken, D. (1996) *The Oracle® designer/2000 handbook*. Boston: Addison-Wesley.

Arango, F. y Zapata, C. M. (2006) *Un-Método para la elicitación de requisitos de software*. Medellín: Carlos Mario Zapata (Ed.).

Beck, K. (2000) *Extreme programming*. Boston: Addison-Wesley.

Chen, P. P. (1976) "The Entity–Relationship Model: Toward a unified view of data". En: *ACM Transactions on DataBase Systems*. Vol. 1. No. 1. New York.

Crhistel, M. y Kang, K. (1992) *Issues in requirements elicitation*. Pittsburg: Software Engineering Institute, Technical Report CMU/SEI-92-TR-12.

Dardenne, A., Van Lamsweerde, A. y Fickas, S. (1993) "Goal-directed requirements acquisition". En: *Science of Computer Programming*. Vol. 20. pp. 3-50.

Fowler, M. (2003) *UML Distilled Third Edition: A brief guide to the standard Object Modeling Language*. Boston: Addison-Wesley.

Gane, C. y Sarson, T. (1986) *Structured system analysis: Tools and techniques. IST Databooks*. St. Louis: MacDonald Douglas Corporation.

Gibbs, W. (1994) "Software's Chronic Crisis". En: *Scientific American*, Septiembre. pp. 72-81.

Grudin, J. (1991) "The Development of Interactive Systems: Bridging the Gaps Between Developers and Users". En: *IEEE Computer.* Vol. 24. No. 4. pp. 59-69.

Harrington, H. J. (1991) *Business process improvement: the breakthrough strategy for total quality, productivity, and Competitiveness*. New York: McGraw-Hill.

Ishikawa, K. (1986) *Guide to quality control*. Tokio: Asian Productivity Organization.

Jacobson, I., Booch, G. y Rumbaugh, J. (1999) *The Unified Software Development Process*. Boston: Addison Wesley Longman.

Omg. (2005) Omg unified modeling language specification. Object Management Group. <From: http://www.omg.org/UML/> (Consulta: 23 de agosto de 2005).

Oracle. (2000) *Oracle Method: CDM Quick tour*. Redwood City: Oracle Corporation.

Sowa, J. F. (1984) *Conceptual structures: information processing in mind and machine*. Reading: Addison-Wesley.

Zapata, C. M. y Arango, F. (2004). "Alineación entre metas organizacionales y elicitación de requisitos del software". En: *Revista Dyna.* No. 143. pp. 101-110.

_____ (2005) "Los modelos verbales en Lenguaje Natural y su utilización en la elaboración de Esquemas Conceptuales para el Desarrollo de software: una revisión crítica". En: *Revista Universidad EAFIT*. Vol. 41. No. 137. pp. 77-95.

Zowghi, D. y Gervasi, V. (2002). "The three cs of requirements: Consistency, Completeness and Correctness". En: *Proceedings of 8th International Workshop on Requirements Engineering: Foundation for Software Quality, (REFSQ'02)*, Essen. pp. 155-164.