

# Capítulo 8

## Un Marco Metodológico para Adoptar los Modelos de Trazado

*La adopción de una nueva teoría, método o técnica debe ir acompañada por una serie de directrices de uso que faciliten su reconocimiento y adopción en ambientes académicos e industriales. En este caso, el desarrollo dirigido por modelos viene creando una nueva forma de pensar y actuar de los ingenieros al enfrentarse al proceso de desarrollo de software. Este nuevo paradigma provee patrones y modelos que estandarizan la transformación de los requisitos a través de las diferentes etapas del ciclo de vida. Por lo tanto, es necesario que su uso sea guiado por una serie de actividades durante el proceso de desarrollo de software.*

*Este capítulo define un marco metodológico que apoya el desarrollo dirigido por modelos soportado por los Modelos de Trazado. Este marco establece un conjunto de actividades que se deben realizar, y diferentes participantes que tienen responsabilidades en los diferentes niveles de refinamiento para lograr un buen nivel en la transformación de los modelos de desarrollo durante el proceso de desarrollo. Así mismo, el marco se apoya en los servicios de trazabilidad que provee el Modelo de Trazado para guiar actividades de control durante la transformación.*

## 8.1 Introducción

El proceso de desarrollo de software se apoya en diferentes modelos o metodologías de desarrollo que incluyen características de definición, modelado, e implementación. Desde el conocido diseño estructurado, pasando por los modelos en cascada, en paralelo, rápido de aplicaciones, prototipos, métodos ágiles, modelo en espiral, el proceso unificado, etc. [Pressman 2006]. Por lo general, los modelos de desarrollo incluyen prácticas de gestión y soporte al proceso que facilitan la obtención de productos confiables y de buena calidad. La gestión está asociada a las actividades del proceso en cada etapa del desarrollo, a los participantes y al grupo de desarrollo, al flujo de acción que determina la metodología, el riesgo, las acciones, los productos y los documentos que se definen finalmente un producto de software.

Los modelos de desarrollo más recientes, como el Proceso de Desarrollo Unificado, complementan la acción de las metodologías estructuradas por medio de la definición de modelos que representan los componentes estáticos y dinámicos del sistema utilizando lenguajes de modelado tales como el UML. En estos modelos cobran vital importancia en la definición de disciplinas y etapas, donde son generados los modelos de desarrollo bajo un proceso iterativo e incremental [Arlow and Neustad 2005]. Actualmente, enfoques tales como SCRUM plantea la combinación de procesos iterativos e incrementales con métodos ágiles de desarrollo donde la generación de productos de software se basa en la conformación de grupos de trabajo especializados, donde la experticia de sus participantes hace posible lograr metas a corto plazo [Schwaber and Beedle 2001].

El desarrollo dirigido por modelos se debe soportar en los modelos en metodologías para que puedan ser adoptados con mayor aceptación en la comunidad académica e industrial actual. Sin embargo, estos son paradigma en proceso de madurez, que aún presenta algunas debilidades en la formalización de las actividades que se deben realizar durante del proceso de desarrollo de tal forma que se pueda adoptar y usar en la industria del software. Este capítulo presenta un marco metodológico que apoya el desarrollo dirigido por modelos soportado en Modelos de Trazado. Por lo tanto se definen actividades, recursos, y entregables que deben gestionarse durante el proceso de desarrollo de software.

La estructura de este capítulo es la siguiente. La Sección 8.1 define el Marco Metodológico. La Sección 8.2 establece las actividades del desarrollo. Finalmente, la Sección 8.3 concluye el capítulo.

## 8.2 Definición del Marco Metodológico

Un marco metodológico se caracteriza por definir una serie de características de gestión que facilitan el seguimiento de actividades y el logro de metas parciales durante un proceso estructurado de negocio, de desarrollo o de construcción de un producto. Un marco metodológico de desarrollo de software provee una estructura bien definida mediante la cual un proyecto de software pueda ser definido, organizado, ejecutado y controlado. Además, proporciona una metodología de trabajo para abordar el desarrollo bajo un modelo específico de desarrollo que facilite la solución de un problema.

El marco metodológico que se define en este capítulo dispone de actividades que se realizan en diferentes niveles de refinamiento para lograr productos durante el proceso de transformación de los modelos de desarrollo. El marco metodológico se soporta en los Modelos de Trazado que pueden especializarse para una gran variedad de sistemas de software proporcionando servicios para la gestión de la configuración de productos de software desarrollados. Figura 107 ilustra la estructura general del marco metodológico para el “Desarrollo dirigido por Modelos Basado en Modelos de Trazado – DOMBET”.

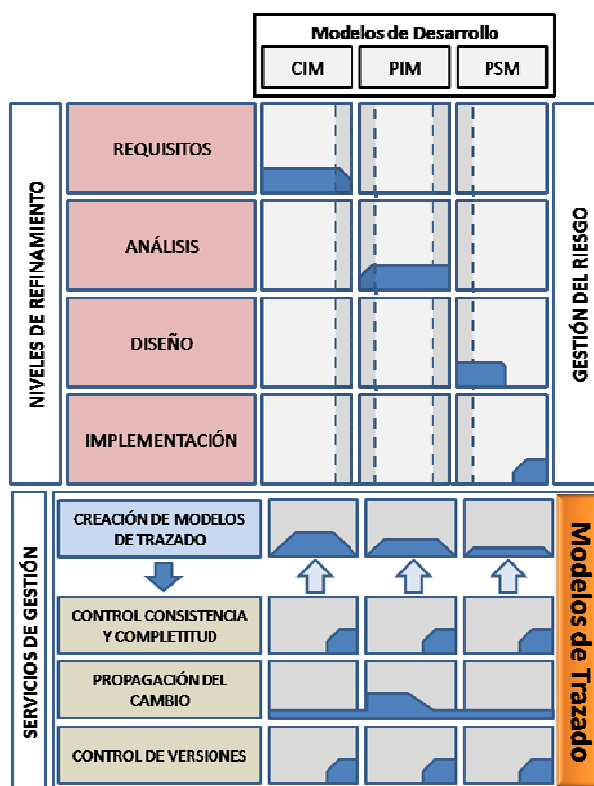


Figura 107. Estructura del marco metodológico DOMBET.

Las características principales de DOMBET son:

**Centrado en la transformación de modelos.** Significa que el desarrollo del software se basa en la transformación de modelos desde los requisitos hasta la implementación.

**Centrado en los Modelos de Trazado.** Significa que los Modelos de Trazado son la herramienta de trazabilidad primaria, subyacente a la especificación del sistema de desarrollo que conduce el proceso de transformación de los modelos del sistema.

**Iterativa e Incremental.** Significa que el sistema se somete al refinamiento a partir de las transformaciones y pruebas continuas durante el ciclo de vida del proyecto.

La estructura de DOMBET está dispuesta de la siguiente forma (de abajo hacia arriba):

- ☞ Servicios de Configuración.
- ☞ Niveles de Refinamiento
- ☞ Modelos de Desarrollo
- ☞ Gestión del Riesgo

### 8.2.1 Servicio de configuración del Proceso de Transformación

Los Modelos de Trazado se han definido como marco conceptual que provee un conjunto de servicios para dirigir las transformaciones de los modelos de desarrollo. Estos servicios son: creación de Modelos de Trazado, control de completitud y consistencia de los modelos resultantes, propagación del cambio, y control de versiones.

**Servicio de Creación de Modelos de Trazado.** Este servicio facilita la creación de Modelos de Trazado que actúan como patrones de control para la transformación de modelos durante el proceso de desarrollo dirigido por modelos. Los Modelos de Trazado son definidos por el grupo de desarrollo y están compuestos de elementos trazables y vínculos de trazado que van a dirigir la transformación de los modelos de desarrollo en diferentes niveles de refinamiento (el detalle de manejo se define en el Capítulo 3).

**Servicio de Control de Consistencia y Completitud.** Una vez que se selecciona el Modelo de Trazado que configura el proceso de transformación, la verificación de la consistencia y la completitud de los modelos de desarrollo se realizarán a medida que nuevos elementos de modelo de un proyecto son creados o actualizados por el grupo de desarrollo. Este servicio se realiza para lograr modelos de calidad en cada nivel de abstracción (el detalle de manejo se define en el Capítulo 3).

**Servicio de Propagación del Cambio.** La gestión del cambio se realiza desde los elementos declarados como ejes del trazado, que normalmente son modelos del nivel PIM. A partir de esto se estima el impacto y el planificador se encarga de ejecutar la propagación del cambio (el detalle de manejo se define en el Capítulo 7).

**Servicio de Control de Versiones.** Cuando se realiza la transformación de modelos y se generan los modelos destino, se crean las instancias versionadas de los modelos de trazados los cuales controlan a su vez la versión de los modelos de desarrollo y sus elementos de modelo. Las vistas versionadas de Modelo de Trazado controla la gestión de cambios en los modelos de desarrollo. Este servicio genera información para herramientas especializadas en este servicio que utilice el equipo de desarrollo (el detalle de manejo se define en el 5).

### 8.2.2 Niveles de Refinamiento

Los niveles de refinamiento son estados que establecen un conjunto de actividades a ser realizadas para el perfeccionamiento de los modelos de desarrollo. DOMBET facilita la labor del equipo de desarrollo al permitirles iniciar la definición de artefactos software en cualquiera de los niveles de refinamiento, y no necesariamente en el nivel de requisitos. No obstante, se sugiere que el equipo de desarrollo defina un orden de trabajo de acuerdo a las actividades sugeridas en cada nivel de refinamiento para cada modelo de desarrollo.

**Requisitos.** Es el nivel de refinamiento donde se crean o se generan por transformación las instancias de los elementos con el rol de predecesor en el Modelo de Trazado seleccionado para coordinar la transformación de modelos. Las tareas realizadas en este nivel definen los modelos de desarrollo CIM.

**Análisis.** Es el nivel de refinamiento donde se crean o se generan por transformación las instancias de los elementos con el rol de ejes del trazado en el Modelo de Trazado seleccionado para coordinar la transformación de modelos. Las tareas realizadas en este nivel definen los modelos de desarrollo PIM. En este nivel el equipo de desarrollo toma decisiones importantes de modelado de acuerdo a las alternativas de modelos generados a partir de la transformación.

**Diseño.** Es el nivel de refinamiento donde se crean o se generan por transformación las instancias de los elementos con el rol de sucesores en el Modelo de Trazado seleccionado para coordinar la transformación de modelos. Las tareas realizadas en este nivel definen los modelos de desarrollo PSM. Es un nivel donde se logran modelos refinados de acuerdo a las decisiones de arquitectura tomadas sobre los modelos generados en nivel PIM. El grupo de desarrollo puede hacer las

iteraciones de refinamiento y transformación que considere necesarias para lograr un buen modelo de diseño.

**Implementación.** Se transforman los modelos PSM hacia el código. Para lograr esto se usan herramientas de transformación que generen código, tomando como fuente los modelos PSM. En caso de que se creen nuevos elementos en este nivel, el grupo de desarrollo debe hacer las transformaciones inversas (*backward*) para mantener la consistencia y completitud de los modelos en los anteriores niveles de refinamiento.

### 8.2.3 Modelos de Desarrollo

En esta metodología cada modelo de desarrollo CIM, PIM, PSM (c.f., Capítulo 3, Sección 3.2.2) constituye un hito de refinamiento en la transformación que determina el nivel de madurez de los modelos. Cada modelo se genera a partir de las actividades de transformación definidas por cada nivel de refinamiento. Para cada uno de ellos se identifican actividades, recursos, información (documentos y modelos) de entrada, y entregables (documento y modelos de salida).

### 8.2.4 Gestión del Riesgo

Por la naturaleza del desarrollo dirigido por modelos, la gestión del riesgo se debe realizar de forma diferente a la que se acostumbra realizar en los procesos de desarrollo normales. La transformación de modelos soportada por los Modelos de Trazado disminuye la incertidumbre relativa a las amenazas generadas por las actividades humanas durante el proceso de desarrollo y las decisiones respecto al manejo de recursos, estrategias de desarrollo, conocimiento de tecnologías, etc.

La definición de los Modelos de Trazado como plantillas de control de la generación de modelos destino, permite identificar los riesgos en cada uno de los niveles de refinamiento de tal forma que el riesgo se pueda transferir a decisiones gerenciales o de conciliación con el patrocinador del proyecto. En otras palabras, los riesgos del nivel operativo que se concentra en los equipos de desarrollo se mitigan de forma sustancial.

Los riesgos están asociados a:

- ☞ La definición de la arquitectura
- ☞ El nivel de exactitud en la definición del Modelo de Trazado.
- ☞ Las decisiones sobre los modelos destino generados por la transformación.

- ☞ La elaboración y refinamiento de documentos de apoyo tales como: documento de visión, documento de factibilidad del proyecto, escenarios de prueba, etc.

### 8.3 Definición de las Actividades de Desarrollo

Cada uno de los niveles de refinamiento provee un conjunto de actividades durante el proceso de transformación para complementar el proceso de transformación que se realiza por medio del planificador de las transformaciones.

Los modelos de desarrollo CIM, PIM y PSM se convierten en productos de software logrados por cada nivel de refinamiento. Las actividades complementarias requieren de recursos humanos que soporten el proceso, ya que las transformaciones se hacen desde el nivel de Requisitos donde se genera el mayor número de alternativas de modelado durante el proceso de transformación de CIM a PIM y viceversa.

El recurso humano que participa en las diferentes actividades se define en la Tabla 34 (recursos sugeridos por el *Eclipse Framework Composer* [Haumer 2007]).

**Tabla 34. Participantes y grupo de desarrollo que son parte de DOMBET.**

<b>Participantes (Stakeholders)</b>	<b>Grupo de Desarrollo</b>
(Usuario) Patrocinador	Gerente del proyecto
(Usuario) Visionario	Arquitecto
(Usuario) Embajador	Analista
	Desarrollador
	Probador ( <i>Tester</i> )

A continuación se definen las actividades y los recursos necesarios en cada nivel de refinamiento, para lograr cada uno de los modelos de desarrollo.

#### 8.3.1 Generación de los Modelos Independientes de la Computación

Un Modelo Independiente de la Computación (CIM) es una representación de componentes del software que refleja todos aquellos elementos que corresponden al modelamiento de las reglas de negocio y requisitos del sistema (modelos de negocio y modelos de dominio). Construir modelos CIM tiene correspondencia con los niveles de refinamiento de Requisitos y Análisis. Estos ayudan a controlar el flujo de las actividades que facilitan la

especificación y realización de los modelos base del sistema. En la Figura 108 se ilustran las actividades generales de que se desarrollan para lograr estos modelos.

**Nivel de Requisitos.** En este nivel se realizan todas las actividades que permiten obtener los modelos fuente de una transformación cuando las actividades de modelado inician en este nivel. Como en cualquier proyecto de desarrollo, para lograr los modelos de la solución primero hay que hacer una abstracción de todo el problema y lograr modelos iniciales de buena calidad.

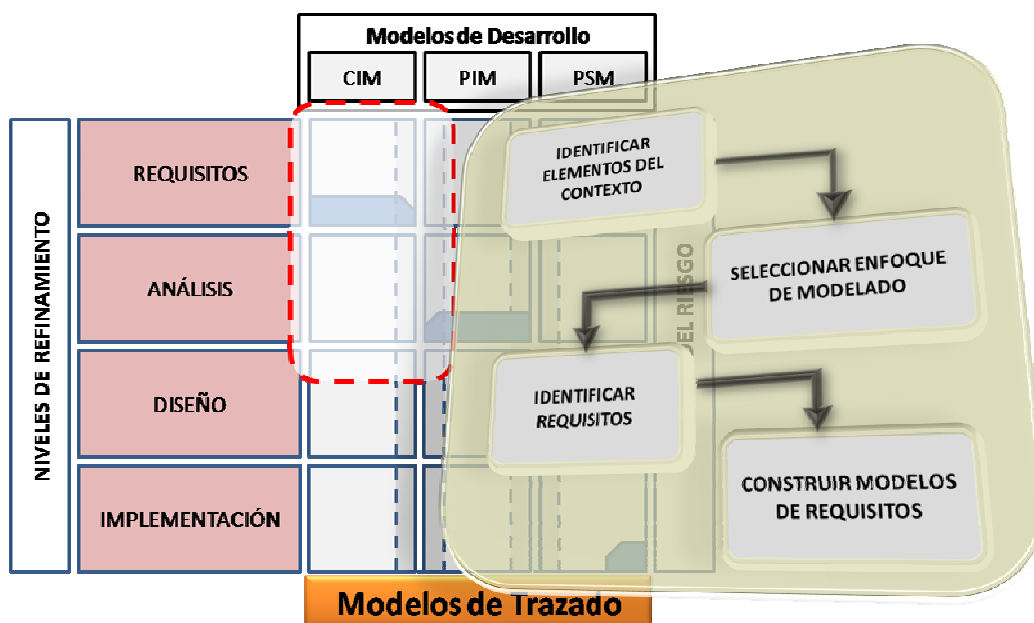


Figura 108. Actividades que hacen parte de la generación de CIM.

☞ **Identificar elementos del contexto del problema.** En esta actividad se definen los elementos generales que determinan la factibilidad de desarrollo del proyecto (económica y técnica), de acuerdo a las necesidades estratégicas del área de negocio y los elementos asociados al dominio del problema. Esta actividad está compuesta por acciones que siguiendo un flujo determinado, guían al equipo de desarrollo en la planeación inicial del proyecto y la identificación de productos de trabajo que formalizan los modelos CIM.

**Recursos de la Actividad.** Patrocinador, Gerente del Proyecto, Visionario, y Arquitecto.

**Acciones.** Definir objetivos, identificar participantes y sistemas externos, elaborar el modelo general de definición del problema, identificar la arquitectura actual, identificar riesgos, aplicar modelo de estimación, realizar la factibilidad del proyecto.

El modelo general del sistema se determina de acuerdo al Modelo de Trazado que se vaya a utilizar para el control del proceso de transformación. Si elige un Modelo de Trazado basado en Casos de Uso, sería un modelo general de casos de uso, de lo contrario sería el modelo de requisitos u otro definido como predecesor o en su defecto eje del trazado si el Modelo de Trazado no considera el nivel de abstracción de los requisitos.

**Información de Entrada.** Necesidades del Negocio, definición formal del problema, arquitectura del sistema actual, catálogo de riesgos. En este nivel, el catálogo de riesgos podría ser el usado comúnmente por el grupo de desarrollo para el tipo de proyecto que se va a desarrollar.

**Entregables.** Definición formal del problema (documento), listado de participantes del proyecto (stakeholders), lista de sistemas externos que interactúan con el sistema, modelo general del sistema, el modelo de la arquitectura actual (lógica y física), la matriz de riesgos (riesgo vs actividad de desarrollo o recurso), y el análisis de factibilidad.

☞ **Seleccionar el enfoque de modelado.** Esta es una actividad básica del proceso. Determina el Modelo de Trazado que controlará el proceso de transformación. La definición de este modelo depende del tipo de aplicación a ser desarrollada, el tamaño del sistema a desarrollar y los riesgos del desarrollo. La selección del enfoque de modelado está compuesta por diversas actividades que siguiendo un flujo determinado, guían al equipo de desarrollo en la obtención del plan de proyecto.

**Recursos.** Gerente del Proyecto, Arquitecto, Analista y Probador.

**Acciones.** Definir objetivos de la aplicación, seleccionar el Modelo de Trazado, seleccionar el Modelo de Desarrollo, refinar modelo de estimación de esfuerzo, y definir el modelo de pruebas.

**Información de Entrada.** Propuesta comercial, análisis de viabilidad técnica, modelos seleccionados.

**Entregables.** Plan de proyecto, Modelo de Trazado, modelo de estimación, matriz de riesgos, modelo de pruebas.

☞ **Identificar requisitos.** Por medio de esta actividad se busca identificar los modelos de requisitos que guiarán el desarrollo del software, partiendo de las definiciones del plan de proyecto, el documento de visión y el modelo de procesos. La identificación de requisitos está compuesta por diversas actividades que, siguiendo un flujo determinado, guían al equipo de desarrollo en la obtención de la especificación formal de requisitos.

Cuando el equipo construye los artefactos partiendo del nivel de refinamiento de los requisitos, por ejemplo el Modelo de Trazado basado en Casos de Uso, se define como punto de partida aquellos modelos que sean considerados predecesores. A partir de las metaclases Requirement, BusinessObject y Class, se crean las instancias de los requisitos, procesos del negocio y modelo del dominio, respectivamente. Si el equipo lo considera conveniente también pueden crear manualmente los casos de uso que son parte de los ejes de trazado, de lo contrario las reglas de transformación de requisitos a casos de uso los generarán. Así, los elementos predecesores hacen las primeras transformaciones hacia los artefactos definidos como ejes de trazado (i.e., casos de uso, clase, e interacción), ya que estos serán los encargados de coordinar los servicios proporcionados por el Modelo de Trazado.

**Recursos.** Arquitecto y Analista.

**Acciones.** Identificar y especificar requisitos, reutilizar catálogos de requisitos, separar o agrupar requisitos.

**Información de Entrada.** Documento de visión, modelo de procesos del negocio, catálogo de intereses, catálogo de dominio, reglas de negocio, catálogos de QoS, documento de requisitos.

**Entregables.** Documento de requisitos.

**Nivel de Análisis.** Existen situaciones en las que el equipo considera pertinente el iniciar la construcción de modelos en un nivel de refinamiento posterior al de requisitos. En este caso, es necesario definir las instancias de los artefactos que son ejes de trazado, permitiendo al patrón evaluar los vínculos de trazado que estén definidos hacia predecesores y sucesores. En la siguiente actividad se puede lograr una primera versión de los modelos CIM y el refinamiento de algunos elementos de modelo del nivel PIM.

- ☞ **Construir modelos de requisitos.** Esta actividad facilita la construcción de los modelos de requisitos que determinen los modelos CIM definitivos, orientando al equipo en el uso de estrategias de modelado. En esta actividad se generan las primeras vistas versionadas de los modelos de trazado (línea base) que permiten la validación de los requisitos y sus transformaciones iniciales a los elementos ejes del trazado y sucesores.

**Recursos.** Arquitecto, Analista y Probador.

**Acciones.** Clasificar requisitos, definir relaciones entre los requisitos, identificar elementos de composición, refinar requisitos, identificar y solucionar conflictos, verificar consistencia y completitud de los modelos, definir escenarios de prueba.

**Información de Entrada.** Documento de requisitos, documento de requisitos.

**Entregables.** Modelo CIM, vista versionada del Modelo de Trazado, reporte de consistencia y completitud inicial, estimación inicial del esfuerzo de desarrollo, matriz de riesgos, y escenarios iniciales de pruebas.

### 8.3.2 Generación de los Modelos Independientes de la Computación

Un Modelo Independiente de la Plataforma (PIM) es una representación de componentes del software que presenta un acercamiento al análisis y diseño de la solución, siendo este, independiente de la plataforma tecnológica de implementación. Podría decirse que son modelos más cercanos a la solución informática, establecida por los modelos CIM. Los elementos declarados como ejes del trazado normalmente forman parte de los modelos PIM. Si consideramos UML como lenguaje de modelamiento, podríamos decir que los modelos son los diagramas de casos de uso, los diagramas de actividades, diagramas de Clases, y cualquier otro que presente la perspectiva dinámica del sistema. En la Figura 109 se ilustran las actividades generales de que se desarrollan en la generación de los PIM.

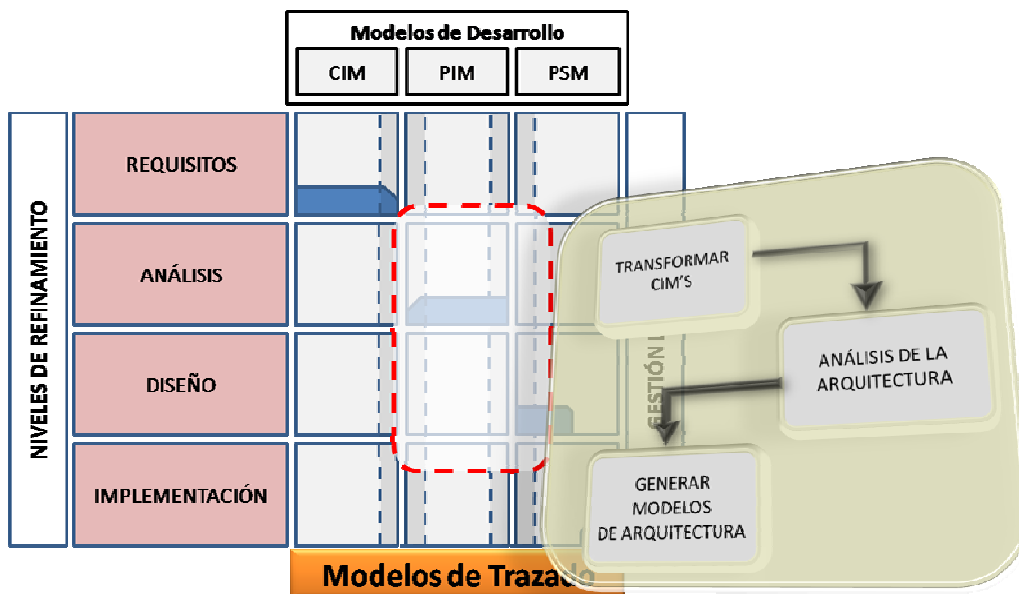


Figura 109. Actividades que hacen parte de la generación de PIM.

**Nivel de Análisis.** En este nivel se realizan todas las actividades que permiten obtener o refinar los elementos “ejes de trazado” de una transformación.

- ☞ **Transformar CIM.** Aunque ya pueden haberse realizado transformaciones, se espera que que en el momento de realizar esta actividad los modelos CIM ya estén definidos o refinados al nivel de garantizar la correcta ejecución de todas las reglas de transformación proporcionadas por el Modelo de Trazado. En este momento se genera una primera versión de los modelos PIM y se crea la vista versionada de Modelo de Trazado que se toma como línea base.

El equipo de desarrollo debe analizar los modelos destino que se han generado, en especial elegir los modelos adecuados cuando la transformación genera más de un modelo destino. Si el grupo de desarrollo lo considera importante, se deben de hacer las correcciones o adiciones de información a los modelos fuente o destino que han sido generados. Las modificaciones realizadas sobre el modelo generado serán controladas por el servicio de propagación del cambio, de lo contrario serán controladas por el servicio de creación de modelos. A partir de los PIM obtenidos, se evalúan los riesgos y se hacen los preparativos necesarios para el análisis y el diseño.

**Recursos.** Analista y Probador.

**Acciones.** Reutilizar catálogos de patrones de análisis, complementar el modelo PIM generado, ejecutar transformaciones necesarias para mantener la consistencia, analizar alternativas de modelado, verificar la completitud de los modelos y los escenarios de prueba.

**Información de Entrada.** Modelo CIM, alternativas de modelado, catálogos de patrones de análisis, última vista versionada del Modelo de Trazado.

**Entregables.** Modelos iniciales PIM, modelo de arquitectura, nueva última vista versionada del Modelo de Trazado, reporte de consistencia y completitud, estimación del esfuerzo, resultados de las pruebas.

- ☞ **Análisis de la arquitectura.** Por medio de esta actividad se busca analizar cuál es la arquitectura más apropiada para el sistema que se encuentra en desarrollo, brindando soporte en catálogos de arquitectura (p.ej., patrones de diseño arquitectónico) y la arquitectura de referencia, acorde al tipo de aplicación (p.ej., web multicapas). Esta actividad provee información al equipo de desarrollo para tomar decisiones sobre los modelos que se tienen actualmente.

El análisis de la arquitectura está compuesto por diversas acciones que, que guían al equipo de desarrollo en la toma de decisiones de arquitectura y la aplicación de patrones que permitirán refinar el modelo inicial PIM. Finalmente se obtiene el documento de arquitectura, la arquitectura de referencia y las decisiones de arquitectura.

**Recursos.** Arquitecto.

**Acciones.** Identificar la arquitectura de referencia, reutilizar catálogos de arquitectura, tomar decisiones de arquitectura, aplicar patrones de arquitectura.

**Información de Entrada.** Modelo PIM, catálogos de arquitectura, arquitecturas históricas de referencia, y decisiones de arquitectura, última vista versionada del Modelo de Trazado.

**Entregables.** Modelos PIM complementados refinados y preparados para la transformación hacia los modelos PSM, última vista versionada del Modelo de Trazado, reporte de consistencia y completitud, estimación del esfuerzo.

**Nivel de Diseño.** En este momento, los modelos PIM tienen un nivel de refinamiento adecuado para transformarlos a PSM. Pero aún así ellos pueden ser complementados con decisiones de arquitectura que permitan realizar una transformación más cercana a los modelos PSM. Es importante resaltar que el Modelo de Trazado a partir de sus vínculos de trazado y secuencias de traza, va generando el modelo PSM de forma temporal que se van refinando a medida que el modelo PIM se ajusta manualmente por parte del analista o arquitecto.

☞ **Generar modelos de arquitectura.** Por medio de esta actividad se busca que el equipo de desarrollo pueda evaluar la pertinencia de reutilizar componentes arquitectónicos predefinidos, complementando el modelo (de ser necesario) antes de ejecutar las reglas de transformación hacia modelos PSM. La generación de modelos de arquitectura está compuesta por diversas actividades que facilitan la verificación de consistencia y completitud sobre el modelo resultante de la transformación, como preámbulo a la aplicación de escenarios de prueba por parte del equipo. Es con base en estos modelos arquitectónicos refinados, que se evalúan los riesgos y se hacen los correctivos necesarios.

**Recursos.** Arquitecto y Probador.

**Acciones.** Reutilizar componentes arquitectónicos, ejecutar reglas de transformación para el refinamiento de los modelos PIM, verificar consistencia y completitud del modelo, verificar escenarios de prueba.

**Información de Entrada.** Modelo PIM.

**Entregables.** Modelo PIM actualizado, última vista versionada del Modelo de Trazado, reporte de consistencia y completitud, estimación del esfuerzo, resultados de las pruebas.

### 8.3.3 Generación de Modelos de Plataforma Específica

Un Modelo de Plataforma Específica (PSM) es una representación de componentes del software que presenta un enfoque de diseño cercano a la implementación en una plataforma tecnológica, llevándolo al nivel de “codificación en un lenguaje específico”. Si consideramos una plataforma de implementación específica, podríamos decir que los modelos PSM se ven representados en la construcción de clases Java o C++, esquemas de bases de datos y asociaciones de intercambio de información. En la Figura 109 se ilustran las actividades generales de que se desarrollan en la generación de los PSM.

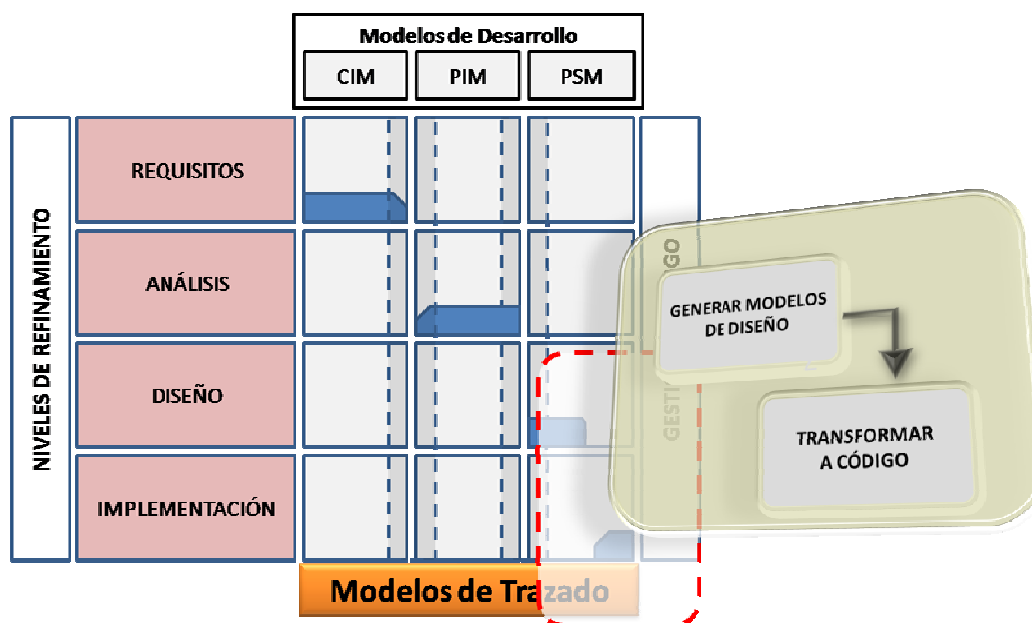


Figura 110. Actividades que hacen parte de la generación de PSM.

**Nivel de Diseño.** En este nivel se realizan todas las actividades de transformación desde los ejes del trazado refinados en los PIM que permiten obtener los modelos sucesores (y potencialmente refinar los predecesores, si es el caso) resultantes de una transformación cuando las actividades de modelado concluyen en este nivel.

- 🌀 **Generar modelos de diseño.** Por medio de esta actividad se busca que el equipo de trabajo pueda evaluar la pertinencia de emplear patrones de diseño para la implementación de la aplicación, adicionando nuevos componentes por medio de la ejecución de reglas de

transformación, y la obtención finalmente los modelos de diseño (correspondientes al modelo PSM).

La generación de modelos de diseño facilita la verificación de la consistencia y completitud sobre el modelo resultante de la transformación, como preámbulo de la aplicación de escenarios de prueba en este nivel, por parte del equipo de desarrollo. Con base en estos modelos de diseño refinados, que se evalúan los riesgos y se hacen los preparativos necesarios para la implementación.

**Recursos.** Arquitecto, Desarrollador y Probador.

**Acciones.** Seleccionar patrón de diseño, ejecutar reglas de transformación, verificar consistencia y completitud del modelo, verificar escenarios de prueba.

**Información de Entrada.** Modelos PSM.

**Entregables.** Modelos PSM refinados para la codificación, última vista versionada del Modelo de Trazado, resultados de las pruebas.

- ☞ **Transformar a código.** Por medio de esta actividad se busca facilitar la ejecución de reglas de transformación que permitan generar código básico para la implementación de la aplicación. La transformación a código está compuesta por diversas actividades que facilitan la verificación de consistencia y completitud sobre el código resultante de la transformación. Es con base en la confrontación del código resultante con los modelos de diseño refinados que se evalúan los riesgos y se hacen los correctivos necesarios.

**Recursos.** Arquitecto, Desarrollador y Probador.

**Acciones.** Ejecutar reglas de transformación, complementar código, verificar consistencia y completitud del modelo, verificar escenarios de prueba.

**Información de Entrada.** Modelo de diseño, modelo de implementación, modelo de arquitectura, última vista versionada del Modelo de Trazado.

**Entregables.** Modelo de implementación, última vista versionada del Modelo de Trazado, estimación del esfuerzo para la implementación, resultados de las pruebas.

## 8.4 Conclusiones

En la actualidad, las metodologías de desarrollo de software se enmarcan en un conjunto de métodos diseñados para disminuir el impacto y costo de los cambios, especialmente en contextos

donde los aspectos vitales surgen a lo largo del proyecto y no al inicio (como se esperaría) o simplemente es necesario realizar adaptaciones posteriores, debido a alteraciones de los requisitos.

Las metodologías del tipo iterativo e incremental [Arlow and Neustad 2005], [Haumer 2007], brindan recursos para la concepción y el modelamiento de las soluciones software que permiten reflejar el dominio del problema a ser resuelto de una forma lo más cercana posible a la realidad del negocio. Sin embargo, solo se obtiene una visión del impacto de los cambios a través de la realización de revisiones formales, básicamente, guiados sobre la premisa que “el costo del cambio es bajo si está correctamente soportado”, esto hace que este tipo de modelos presenten la trazabilidad como una necesidad importante pero aislada, ya que no muestran la manera en la que se debe llevar a cabo, quienes serían los responsables y el impacto que esto tendría en el proyecto.

DOMBET, como marco metodológico va más allá de la simple aplicación de ciclos iterativos en el desarrollo de software. Las iteraciones dependen del patrón de transformación proporcionado por el desarrollo dirigido por procesos, en este caso el que se determina por medio del Modelo de Trazado utilizado durante el desarrollo. Este complemento brinda al proceso características que facilitan la labor de evolución de los modelos construidos como base del desarrollo de software, permitiendo a los grupos de desarrollo mantener control constante del proceso de transformación (que anteceden a los cambios evolutivos en los modelos) y hacer su respectivo seguimiento, de forma tal que el proceso de trazabilidad sea menos complejo y más completo que la simple indicación de la necesidad de hacerlo, que es lo que realmente hacen los otros modelos de desarrollo de software.

DOMBET se concibe como un marco de trabajo enriquecedor para los equipos de desarrollo, ya guía el uso de una metodología iterativa e incremental, en convivencia con un desarrollo dirigido por modelos. Además, determina las responsabilidades que tienen los roles frente a las actividades propias de este tipo de desarrollo, especialmente aquellas que se relacionan directamente con el manejo de la trazabilidad. Esto puede verse reflejado en actividades tales como: la transformación de los modelos, la aplicación de patrones (ya sea de diseño, arquitectura o requisitos), y la gestión constante del riesgo (por medio de la definición y aplicación de escenarios de prueba).

Considerando la evolución de las metodologías de desarrollo de software hacia el enfoque de modelos, DOMBET brinda grandes ventajas a los equipos de trabajo, ya que no solo sigue los lineamientos de concepción y construcción del software dirigido por modelos, sino que los complementa apoyándose en los Modelos de Trazado, como patrones de control del desarrollo de un proyecto de software. Esto fortalece los criterios de calidad del producto a través del aseguramiento de la calidad del proceso.

En la actualidad se analiza la estrategia de implantar DOMBET en la industria del software, comparando su aplicación en proyectos implementados bajo modelos típicos en metodologías iterativas e incrementales. Para lograr esto, se considera la herramienta *Eclipse Process Framework Composer* como apoyo para formalizar la gestión de los proyectos definidos bajo este marco de trabajo. Esta herramienta brinda soporte a la gestión del proceso de desarrollo, facilitando su estandarización y difusión por toda la organización [Haumer 2007].

Este capítulo fue realizado con el apoyo de los participantes del grupo de investigación en Ingeniería de Software de la Universidad EAFIT y la empresa de desarrollo de software AVANSOFT.

# Capítulo 9

## Conclusiones y Trabajo Futuro

*Este capítulo revisa los objetivos de esta tesis de doctorado, discute como estos fueron trabajados y logrados. Además argumenta acerca de los posibles trabajos futuros que esta tesis genera en el campo de la investigación aplicada.*

### 9.1 Resumen de los objetivos de la tesis

El objetivo de esta tesis de doctorado fue orientado al tratamiento de la trazabilidad desde el desarrollo dirigido por modelos con aplicación especial para los asuntos definidos bajo el dominio del desarrollo de software orientado por aspectos. En especial, la trazabilidad se definió como mecanismo para lograr consistencia y completitud de los modelos de desarrollo desde los requisitos, a través del análisis, hasta lograr la arquitectura de diseño.

Finalmente, se lograron combinar ideas de la trazabilidad de requisitos, y de los dos paradigmas de desarrollo el MDD y el AOSD para lograr los siguientes objetivos específicos:

- ❧ Identificar diferentes características, elementos y comportamiento que requiere un modelo de trazabilidad para soporte la evolución y transformación de los asuntos y asuntos transversales en diferentes niveles de abstracción.
- ❧ Definir un mecanismo para lograr la verificación de la consistencia y la completitud de los asuntos durante la transformación de los modelos.
- ❧ Perfilar un método para realizar la propagación del cambio y medir su impacto desde la transformación de modelos.

- ☞ Definir diferentes tipos de asuntos que puedan representar los objetivos del sistema como un todo de tal forma que este se pueden trazar y transformar en diferentes niveles de abstracción.

## 9.2 Resultados de la Tesis

El trabajo realizado en esta tesis contribuye directamente a la práctica de la trazabilidad de los requisitos a través de diferentes niveles de abstracción. Específicamente se define la trazabilidad un patrón de control definido al nivel meta que facilita las prácticas de validación y verificación que se realiza reiterativamente durante el proceso de desarrollo dirigido por modelos.

Inicialmente en el Capítulo 2 se hace un estudio profundo de la literatura de investigación que aporta al conocimiento del estado de la técnica y su aporte directo al logro de consistencia y completitud de modelos desde la práctica de la trazabilidad.

A partir de los escenarios de trabajo identificados, en el Capítulo 3 se crea el metamodelo llamado “Patrón de Trazabilidad” para estandarizar la creación y gestión de elementos trazables y vínculos de trazado por medio de los Modelos de Trazado que garantizan confiabilidad al modelar y especificar cualquier tipo de artefacto software, ya sean los requisitos, elementos de modelo al nivel de análisis y diseño o asuntos en diferentes niveles de abstracción.

Los Modelos de Trazado constituyen un marco conceptual que caracteriza la práctica de la trazabilidad y la transformación de modelos para cualquier entorno de desarrollo. El marco conceptual, en primera instancia estandariza el uso de los vínculos de trazado definidos en UML con el fin de controlar la transformación de modelos. Esto elimina una de las grandes incertidumbres de los desarrolladores al usar una relación de dependencia para la trazabilidad, ya que sus definiciones son tan similares que no es fácil identificar la situación real de uso, además de la utilidad de usar los diferentes tipos ofrecidos por UML.

La declaración de los Modelos de Trazado se hace al nivel del metamodelo. Esto facilita su utilización como patrones dinámicos que estandarizan y guían la práctica de la trazabilidad de los proyectos de software a partir de los criterios de calidad, el tipo de proyecto o las políticas de desarrollo más usadas por los grupos de trabajo en la industria del software. En esta disertación se definen dos tipos de Modelo de Trazado: basados en Casos de Uso y basados en Modelos de Asuntos (en el Capítulo 5). El primero se orienta a que el uso de estos modelos pueda ser adoptado con facilidad por la industria del software que normalmente modela artefactos UML y usa el Proceso de Desarrollo Unificado [Arlow and Neustad 2005]. El segundo, está orientado a

soportar la trazabilidad de asuntos [Tabares et al. 2008]. Los Modelos de Trazado facilitan tareas o servicios de trazabilidad y transformación de modelos tales como:

- ☯ La realización de trazabilidad horizontal y vertical por medio de roles asignados a los elementos trazables de un Modelo de Trazado. Los vínculos de trazado definidos entre elementos identificados por un mismo rol, garantizan la trazabilidad vertical (intra-nivel) y las trazas definidas entre elementos de diferentes roles garantiza la trazabilidad horizontal. Esto facilita la verificación de la consistencia y la completitud de los modelos.
- ☯ La trazabilidad bidireccional por medio de los vínculos de trazado y las reglas de transformación en doble sentido asociadas a estos. Esto garantiza el control de predecesores y sucesores generados durante el proceso de transformación.

La especificación de un mecanismo de transformación de modelos se hace en el Capítulo 4 para facilitar la ejecución de diferentes tipos de reglas que son asociadas a los vínculos de trazado de diferentes Modelos de Trazado. Este permite además:

- ☯ La identificación elementos que pueden ser parte de los modelos de la fuente y los modelos del destino de una regla de transformación.
- ☯ La creación y mantenimiento reglas de transformación a partir de los meta elementos definidos por el Patrón de Trazabilidad.
- ☯ La definición de diferentes tipos des reglas de transformación en un lenguaje natural y declarativo para facilitar la operacionalización de la trazabilidad.
- ☯ El uso dinámico de las reglas de transformación de acuerdo a las necesidades de consistencia y completitud de los modelos, el manejo de alternativas de transformación para un mismo modelo fuente, cuando sea necesario
- ☯ La planificación de la ejecución de las reglas de transformación que facilite servicios agregados de trazabilidad tales como la consistencia y la completitud y el manejo del cambio.

Además, la trazabilidad por Modelos de Trazado responde también al siguiente interrogante: ¿Es posible medir el esfuerzo que requiere la gestión del cambio durante la transformación de modelos? Para responder a esta, en el Capítulo 7 se define la gestión del cambio con las siguientes características:

- ☯ Se define un método de trazabilidad que facilita la definición de Modelos de Trazado y la gestión del cambio.

- ☯ Se definen una plantilla que formaliza la gestión de los requisitos de cambio que tanto los usuarios como los grupos de desarrolladores podrían definir sobre los modelos de desarrollo existentes.
- ☯ Se clasifican los requisitos de cambio de acuerdo a las operaciones de cambio que se deba realizar sobre los modelo fuente (del cambio).
- ☯ Se establece un procedimiento de ejecución de cada una de las operaciones de cambio: crear, modificar y eliminar, desde las vistas versionadas de Modelos de Trazado.
- ☯ Se define estimación del impacto del cambio a partir de las vistas versionadas de Modelo de Trazado definido para el sistema en desarrollo. La estimación se hace con base en las métricas de transformación definidas en el Capítulo 4.
- ☯ Se define la propagación del cambio en los modelos de desarrollo definidos para el sistema en desarrollo, a partir de las vistas versionadas de Modelo de Trazado.

De esta forma se define el valor agregado que se tiene realizar la práctica de la trazabilidad desde el contexto del desarrollo de software dirigido por modelos. No obstante, el paradigma MDD o MDE aún sigue en proceso de madurez tanto conceptualmente como al nivel de herramientas de soporte.

Especialmente, en lo que se refiere la transformación de los CIM hacia PIM y PSM, esta disertación hace un gran aporte en el Capítulo 4 al definir la identificación, separación y tipificación de los asuntos desde el nivel de los requisitos, y así lograr características de modificabilidad, consistencia y completitud en el espacio multidimensional que definen los Modelos de Asuntos. Esto es posible al lograr:

- ☯ La identificación, descomposición y clasificación de asuntos y sus requisitos. Además, cinco tipos de asuntos se definen al nivel del Patrón de Trazabilidad para crear elementos cuyas instancias se clasifican dinámicamente.
- ☯ La creación de relaciones de dependencias orientadas hacia la trazabilidad entre asuntos que describen un objetivo del sistema.
- ☯ El control del impacto del cambio y su propagación en asuntos de diferentes niveles de abstracción de acuerdo con su definición en los Modelos de Trazado.

Finalmente, en el Capítulo 8 se logra constituir un marco metodológico que promueve la adopción y uso tanto de los Modelos de Trazado como de los Modelos de Asuntos. Este marco metodológico facilita actividades tales como:

- ☞ El establecimiento de los Modelos de Trazado como el marco conceptual que soporta la gestión de la configuración para el desarrollo dirigido por modelos.
- ☞ La identificación de actividades y recursos que facilitan o contribuyen a la gestión de proyecto de desarrollo dirigidos de modelos.

### **9.3 Trabajo Futuro**

El trabajo futuro se centra en las siguientes áreas de desarrollo de investigación aplicada:

1. Inclusión de servicios de trazabilidad en una herramienta CASE de trazabilidad soportados en el Patrón de Trazabilidad y acorde a la forma como se planteó el estudio de caso del Capítulo 5 y el marco metodológico definido en el Capítulo 8. De tal forma que se convierta en el marco de trabajo que soporte el modelo de trazabilidad conceptual y metodológico definido en esta tesis.
2. La implantación del marco metodológico en al menos dos empresas de desarrollo de software de la ciudad de Medellín. Esto requerirá de tiempo de capacitación en la definición y uso de los modelos de trazado así como la adopción de la cultura de la transformación de modelos, ya que las herramientas CASE que son utilizadas con frecuencia por estas empresa no hacen transformación de CIM a PIM.
3. La aplicación de los Modelos de Trazado en proyectos que requieran la gestión del cambio orientada a lograr consistencia y completitud en en la gestión de modelos. Actualmente se está utilizando este marco conceptual para el análisis de la gestión de los cambios que ocurren en los modelos del negocio y el impacto en el proceso de desarrollo, y viceversa.
4. Aplicar los Modelos de Asuntos a diferentes tipos de proyectos de software. En este momento se trabaja en la aplicación de estos modelos en un proyecto de investigación en Telesiquiatría, donde la separación de asuntos es la base del desarrollo y aplicación.

# Referencias

---

- Agrawal, A., Karsai, G., & Shi, F. (2003). Graph Transformations on Domain-Specific Models. *Technical Report ISIS-03-403 of the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37203*.
- Ahn, S., & Chong, K. (2006). A Feature-Oriented Requirements Tracing Method: A Study of Cost-benefit Analysis. *International Conference on Hybrid Information Technology. ICHIT*, 2(9-11), 611 – 616.
- Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., & Shaham-Gafni, Y. (2006). Model traceability. *IBM Systems Journal*, 45(3).
- Aizenbud-Reshef, N., Paige, R. F., Rubin, J., Shaham-Gafni, Y., & Kolovos, D. S. (2005). *Operational Semantics for Traceability*. Nuremberg, Germany: EC-MDA Traceability workshop.
- Albrecht, A.J. (1979). Measuring Application Development Productivity. In Proc. IBM Application Development Symposium, Monterrey, CA. Octubre, 1979. 83-92.
- Alencar, F. M. R., Silva, C. T. L. L., Lucena, M., Brelaz de Castro, J., Santos, S., & Ramos, R. (2008). Improving the Understandability of I\* Models. En ICEIS (Vol. 3, págs. 129-136).
- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1989). A pattern language. *Oxford University Press*.
- Almeida, J. P., Eck, P. V., & Iacob, M. E. (2006). Requirements Traceability and Transformation Conformance in Model-Driven Development. *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*.
- Amar, B., Leblanc, H., & Coulette, B. (2008). *A Traceability Engine Dedicated to Model Transformation for Software Engineering*. ECMDA Traceability Workshop (ECMDA-TW) Proceedings.

- Arlow, J., & Neustad, I. (2005). *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design* (2° ed.). Addison-Wesley Object Technology Series.
- Azelborn, B. (2000). *Building a Better Traceability Matrix with DOORS*. Telelogic INDOORS US.
- Bachmann, F., Bass, L., & Nord, R. (2007). Modifiability Tactics. *Software Engineering Institute Technical Report CMU/SEI-2007-TR-002, ESC-TR-2007-002*.
- Baniassad, E., & Clarke, S. (2004). Finding Aspects in Requirements with Theme/Doc. *In Workshop on Early Aspects*.
- Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2° ed., págs. 105-111). Addison Wesley.
- Berg, v. d. K., Tekinerdogan, B., & Nguyen, H. (2006). Analysis of Crosscutting in Model Transformations. En *ECMDA-TW Traceability Workshop Proceedings* (págs. 51-64). Bilbao, Spain.
- Bezivin, J., Jouault, J., & Touzet, D. (2005). *An Introduction to the ATLAS Model Management Architecture*. Research Report, LINA (05-01).
- Bloomfield, T. (2005). MDA, Meta-Modelling and Model Transformation: Introducing New Technology into the Defence Industry (págs. 9-18). ECMDA-FA.
- Boehm, B. W. (1984). Verifying and Validating Software Requirements and Design Specifications. *IEEE Software*, 1(1), 75-88.
- Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. J., Steece, B., (2000). *Software Cost Estimation in COCOMO II*. Prentice-Hall. 2000.
- Bohner, S. A. (2002). Software Change Impacts – An Evolving Perspective. En *Proceedings of the International Conference on Software Maintenance (ICSM'02)*. IEEE Computer Society.
- Bondé, L., Boulet, P., & Dekeyser, J. (2005). Traceability and interoperability at different levels of abstraction in model transformations. En *in Forum on Specification and Design Languages, FDL'05*. Lausanne, Switzerland.

- Briand, L. C., Labiche, Y., & O’Sullivan, L. (2003). Impact Analysis and Change Management of UML Models. En *Proceedings of the International Conference on Software Maintenance (ICSM’03)*.
- Brito, I. S. (2008, Julio). *Aspect-Oriented Requirements Analysis*. Departamento de Informática, FCT, Universidade Nova de Lisboa.
- Brown, A. (2004). *An introduction to Model Driven Architecture, Part I: MDA and today's systems*. The Rational Edge.
- Carroll, E. R. (2005). Estimating Software Based on Use Case Points. En *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (págs. 257-265). San Diego, CA, USA: ACM.
- Chen, P. P. (1985). “Database Design Using Entities and Relationships,” in: S. B. Yao (ed.), *Principles of Data Base Design* (págs. 174-210). Prentice-Hall.
- Chitchyan, R., Pinto, M., Rashid, A., & Fuentes, L. (2007). COMPASS: Composition-Centric Mapping of Aspectual Requirements to Architecture. En *Transactions on Aspect-Oriented Software Development IV* (Vol. 4640, págs. 3-53). Springer.
- Chitchyan, R., Rashid, A., Rayson, P., & Waters, R. W. (2007). Semantics-based Composition for Aspect-Oriented Requirements Engineering (págs. 36-48). Vancouver, Canada: ACM.
- Chitchyan, R., Rashid, A., Sawyer, P., Bakker, J., Pinto, M., García, A., et al. (2005). Survey of Aspect-Oriented Analysis and Design Approaches. AOSD-Europe-ULANC-9, AOSD-EUROPE network of excellence.
- Clarke, S. (2002). Extending standard UML with model composition semantics. in *Science of Computer Programming*, 44(1), 71-100.
- Clarke, S., & Baniassad, E. (2005). *Aspect-Oriented Analysis and Design. The Theme Approach*. Addison-Wesley, Object Technology Series.
- Cleland-Huang, J., Chang, C. K., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *TSE*, 29(9), 796 – 810.
- CMMI® for Development, Version 1.2. (2006, Agosto). Carnegie Mellon – Software Engineering Institute. *CMU/SEI-2006-TR-008. ESC-TR-2006-008*.

- Conejero, J. M., Berg, K., & Hernandez, J. (2007). Un marco conceptual para la formalización de crosscutting en desarrollos orientados a aspectos. *IEEE Latin America Transactions*, 5(2).
- Csertan, G., Huszerl, G., Majzik, I., Pap, Z., Pataricza, A., & Varro, D. (2002). VIATRA - visual automated transformations for formal verification and validation of UML models. *Automated Software Engineering, 2002. Proceedings. ASE 17th IEEE International Conference*, 267 – 270.
- Czarnecki, K., & Helsen, S. (2006). Feature-based Survey of Model Transformation Approaches. *IBM Systems Journal*, 45(3), 621–646.
- Deaño, A. (1975). *Introducción a la lógica formal. 2. Lógica de predicados* (págs. 200-201). Madrid: Alianza Eds.
- Derezińska, A., & Zawłocki, J. (s.d.). Application of Model Transformation in the Generic Framework for Traceability. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 715 – 724.
- Dijkstra, E. W. (1976). *A Discipline of Programming*. Englewood Cliffs, NJ: Prentice Hall.
- Drivalos, N., Paige, R. F., Fernandes, K. J., & Kolovos, D. S. (2008). *Towards Rigorously Defined Model-to-Model Traceability*. ECMDA-Traceability.
- Egyed, A. (2002). Automated Abstraction of Class Diagrams. *ACM Transactions on Software Engineering and Methodology*, 11(4), 449–491.
- Egyed, A. (2003). A scenario-driven approach to trace dependency analysis. *IEEE Transactions on Software Engineering*, 29, 116-132.
- Egyed, A. (2004). Resolving uncertainties during trace analysis. En *Proceedings of the 12th ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE)* (págs. 3-12). Irvine, CA: ACM.
- Egyed, A. (2006). Instant Consistency Checking for the UML. En *Proceedings of the 28th international conference on Software engineering* (págs. 381-390). Shanghai, China: ACM.
- EMOF. (2006). *Meta Object Facility (MOF) Core Specification*. OMG Available Specification Version 2.0, formal/06-01-01.

- Engels, G., Küster, J. M., Heckel, R., & Groenewegen, L. (2002). Towards consistency-preserving model evolution. En *Proceedings of the International Workshop on Principles of Software Evolution* (págs. 129-132). New York: ACM Press.
- Falleri, J., Huchard, M., & Nebut, C. (2006). Towards a Traceability Framework for Model Transformations in Kermeta. En *ECMDA Traceability Workshop (ECMDA-TW)* (págs. 31-40).
- Feng, Y., Huang, G., Yang, J., & Mei, H. (2006). Traceability between Software Architecture Models. *Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International*, 2, 41-44.
- Filman, R. E., Elrad, T., Clarke, S., & Aksit, M. (2004). *Aspect-Oriented Software Development*. Addison-Wesley.
- Finkelstein, A., & Sommerville, I. (1996). The Viewpoints FAQ. *BCS/IEE Software Engineering Journal*, 11(1), 2-4.
- Galvão, I., & Goknil, A. (2007). Survey of Traceability Approaches in Model-Driven Engineering. *Proceedings of the Eleventh IEEE International EDOC Enterprise Computing Conference*, 313-324.
- Galvão, I., Khan, S. S., Noppen, J., Royer, J., Rummler, A., Sánchez, P., et al. (2008). Definition of a traceability framework (including the metamodel and the modelling of processes and artifacts to allow traceability in the presence of uncertainty) for SPLs. En *AMPLE Aspect-Oriented, Model-Driven, Product Line Engineering Specific Target Research Project: IST-33710*.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, Inc.
- Gotel, O., & Finkelstein, A. C. W. (1994). An analysis of the Requirements Traceability Problem. In: *Proc. 1st IEEE Int. Conf. on Requirements Engineering*, 94-101.
- Gotel, O., & Finkelstein, A. C. W. (1997). Extended requirements traceability: results of an industrial case study. *3rd IEEE Intl. Symposium on Requirements Engineering*.
- Grills, M. (2005). Survey of traceability models in IT projects. En *Proceedings ECMDA Traceability Workshop (ECMDA-TW)* (págs. 39-46).

- Gruenbacher, P., Egyed, A., & Medvidovic, N. (2000). Dimensions of Concerns in Requirements Negotiation and Architecture Modeling. *Proceedings of the 2nd Workshop on Multi-Dimensional Separation of Concerns (MDSOC) in ICSE 2000*.
- Grundy, J. (1999). Aspect-Oriented Requirements Engineering for Component-based Software Systems. *Proceedings of the 4th IEEE International Symposium on Requirements Engineering*, 84-91.
- Grundy, J. C., Mugridgett, W. B., & Hosking, J. G. (1998). Inconsistency Management for Multiple-View Software Development Environments. *IEEE Transactions on Software Engineering*, 24(11), 960-981.
- Heindl, M., Biffel, S., & Egyed, A. (2006). Requirements Tracing Strategies for Change Impact Analysis and Re-Testing. En *International Symposium on Empirical Software Engineering*.
- IBM Rational® RequisitePro®. (2008). Rational RequisitePro. <http://www-306.ibm.com/software/awdtools/reqpro/>.
- IEEE, E. (1991). IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. *Institute of Electrical and Electronics Engineers*.
- Jackson, A., Sánchez, P., Fuentes, L., & Clarke, S. (2006). Towards Traceability between AO Architecture and AO Design. En *Proc. of the 8th Workshop on Early Aspects (EA), 5th Int. Conference on Aspect-Oriented Software Development (AOSD)*. Bonn, Germany.
- Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *The Unified Software Development Process*. Addison Wesley.
- Jacobson, I., & Ng, P. (2005). *Aspect-Oriented Software Development with Use Cases*. Addison Wesley Professional.
- Jouault, F., & Kurtev, I. (2005). Transforming Models with ATL. In *Proc. of the Model Transformations in Practice Workshop at MoDELS*.
- Kalnins, A., Barzdins, J., & Celms, E. (2004). Model transformation language MOLA. In: *Proc. Model-Driven Architecture: Foundations and Applications*, 14-28.
- Kande, M. M. (2003). *A Concern-oriented approach to software architecture*. In *Computer*

- Science, vol. PhD, Swiss Federal Institute of Technology (EPFL).
- Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & Griswold, W. G. (2001). Getting started with AspectJ. *Comm. of the ACM*, 44(10), 59-65.
- Kleppe, A., Warmer, J., & Bast, W. (2003). *MDA Explained, The Model-Driven Architecture: Practice and Promise*. Addison Wesley.
- Koch, N. (2006). Transformation Techniques in the Model-Driven Development Process of UWE. *Workshops at ICWE*.
- Koteswara Rao, K., Madhusudhana Rao, T. V., Sumender Roy, M., Sagar Reddy, G. S., Sankar Sharath, S. S., Kuma S, A., et al. (2008). Effort Estimations Based on Lines of Code and Function Points in Software Project Management. *IJCSNS International Journal of Computer Science and Network Security*, 8(6).
- Kulkarni, V., & Reddy, S. (2003). Separation of Concerns in model-driven development. *Software, IEEE*, 20(5), 64-69.
- Kurtev, I., Dee, M., Goknil, A., & Berg, K. (2007). Traceability-based Change Management in Operational Mappings. En *Proceedings ECMDA Traceability Workshop (ECMDA-TW)*.
- Lau, K. (2004). *Component-Based Software Development: Case Studies (Series on Component-Based Software Development)*. Word Scientific Publishing.
- Lawley, M., & Steel, J. (2005). Practical Declarative Model Transformation with Tefkat. En *Proceedings of Model Transformations in Practice Workshop, MoDELS Conference*. Montego Bay, Jamaica.
- Lindvall, M., & Sandahl, K. (1996). Practical implications of traceability. *Journal of Software Practice and Experience*, 26(10), 1161-1180.
- MDA-Guide. (s.d.). OMG Document V1.0.1. omg/03-06-01. [www.omg.org](http://www.omg.org).
- Mens, T., Czarnecki, K., & Van Gorp, P. (2006). A Taxonomy of Model Transformation. *Electr. Notes Theor. Comput. Sci.*, 152(27), 125-142.
- Mens, T., Van Gorp, P., Varró, D., & Karsai, G. (2006). Applying a Model Transformation Taxonomy to Graph Transformation Technology. *Electr. Notes Theor. Comput.*

- Sci.*, 152(27), 143-159.
- MOF-OMG. (2006). Meta Object Facility (MOF) Core Specification. *OMG Available Specification Version 2.0*.
- MOF-QVT. (2005). Object Management Group, MOF QVT Final Adopted Specification. *OMG Adopted Specification ptc/05-11-01*.
- Moreira, A., Rashid, A., & Araújo, J. (2005). Multi-Dimensional Separation of Concerns in Requirements Engineering. En *The 13th International Conference on Requirements Engineering* (págs. 285 - 296).
- Moreira, A., Araújo, J., & Rashid, A. (2005). A Concern-Oriented Requirements Engineering Model. En *Advanced Information Systems Engineering, Lecture Notes in Computer Science* (Vol. 3520, págs. 293-308). Springer.
- Moreira, A., Araújo, J., & Whittle, J. (2006). Modeling volatile concerns as aspects. En *Proceedings in 18th international conference, Advanced Information Systems Engineering, Lecture Notes in Computer Science* (Vol. 4001, págs. 544-558). Luxembourg: Springer.
- Nuseibeh, B., Easterbrook, S., & Russo, A. (2000). Leveraging Inconsistency in Software Development. *IEEE Computer*, 33(4), 24-29.
- Olsson, T., & Grundy, J. (2002). Supporting Traceability and Inconsistency Management Between Software Artifacts. En *Proceedings of the IASTED International Conference on Software Engineering and Applications*.
- OMG-CWM. (2003, Marzo). Common Warehouse Metamodel (CWM) Specification. Formal document.
- Ospina, C., Parra, C., Londoño, L., & Anaya, R. (2007). Extensión al modelo de separación multidimensional de intereses en Ingeniería de Requisitos. En *Memorias del X Workshop en Ingeniería de Requisitos y Ambientes Software (IDEAS)*.
- Paige, R. F., Olsen, G. K., Kolovos, D. S., Zschaler, S., & Power, C. (2008). *Building Model-Driven Engineering Traceability Classifications*. ECMDA Traceability Workshop (ECMDA-TW).
- Parnas, D. L. (1972). *On the criteria to be used in decomposing systems into modules* (12°

- ed., Vol. 15, págs. 1053 - 1058). ACM.
- Parthasarathy, M. A. (2007). *Practical Software Estimation: Function Point Methods for Insourced and Outsourced Projects*. Infosys Press.
- Pérez, F. J., Laguna, M. A., Crespo, Y., & González-Baixauli, B. (2006). Requirements variability support through MDD and graph transformation. In: *International Workshop on Graph and Model Transformation (GraMoT'05)*, Electronic Notes in Theoretical Computer Science., 152, 161-173.
- Pfleeger, S. L., & Atlee, J. M. (2005). *Software Engineering: Theory and Practice* (3° ed.). Prentice Hall.
- Pinto, M., & Fuentes, L. (s.d.). AO-ADL: An ADL for Describing Aspect-Oriented Architectures. En *Early Aspects: Current Challenges and Future Directions* (Vol. 4765, págs. 94-114). Springer.
- Pressman, R. S. (2006). *Ingeniería del Software: Un enfoque práctico* (6° ed.). McGraw Hill.
- Ramesh, B. (1998). Factors influencing requirements traceability practice. En *Communications of the ACM* (Vols. 1-12, Vol. 41, págs. 37-44). ACM.
- Ramesh, B., & Jarke, M. (2001). Towards reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 27(1).
- Rashid, A., Moreira, A., & Araújo, J. (2003). Modularization and composition of Aspectual Requirements. In Proc. *Conference on Aspect-Oriented Software Development*. Boston, USA.
- Reddy, Y. R., Ghosh, S., France, R. B., Straw, G., Bieman, J. M., McEachen, N., et al. (2006). Directives for Composing Aspect-Oriented Design Class Models. En *Transactions on Aspect-Oriented Software Development I* (Vol. 3880, págs. 75-105). Springer.
- Rensink, A. (2004). The GROOVE simulator a tool for state space generation. En *Applications of Graph Transformations with Industrial Relevance (AGTIVE)*, *Lecture Notes in Computer Science* (Vol. 3062, págs. 479–485). Springer.
- Robins, J., & and others. (s.d.). ArgoUML. <http://argouml.tigris.org/>. Refernciado

[31(08/2006)].

- Saeki, M., & Kaiya, H. (2007). Measuring Model Transformation in Model Driven. En CAiSE'07 Forum Proceedings (págs. 77-80).
- Schneider, G., & Winters, J. P. (2001). *Applying Use Cases: A Practical Guide* (2° ed.). Reading, MA: Addison-Wesley.
- Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall.
- SE Tools Taxonomy - Requirements Traceability Tools. (2004, Septiembre 22). International Council on Systems Engineering.
- Solberg, A., Simmonds, D., Reddy, R., Ghosh, S., & France, R. (2005). Using Aspect Oriented Techniques to Support Separation of Concerns in Model Driven Development. En *29th Annual International Computer Software and Applications Conference (COMPSAC)*. Volume 1, Issue, 121 - 126 Vol 2.
- Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering - A Good Practice Guide*. John Wiley and Sons.
- Sousa, K., Mendonça, H., Vanderdonckt, J., Rogier, E., & Vandermeulen, J. (2008). User Interface Derivation from Business Processes: A Model-Driven Approach for Organizational Engineering. En *Proceedings of the ACM symposium on Applied computing*.
- Sprinkle, J., Agrawal, A., Levendovszky, T., Shi, F., & Karsai, G. (2003). Domain model translation using graph transformations. In: *Proc. Int'l Conf. Engineering of Computer-Based Systems*. Págs: 159–168.
- Stahl, T., & Volter, M. (2006). *Model-Driven Software Development-Technology, Engineering, Management*. Chichester, England : John Wiley and Sons, Ltd.
- Steinberg, D., Budinsky, F., Paternostro, M., & Merks, E. (2008). *EMF: Eclipse Modeling Framework* (2° ed.). Addison Wesley Professional.
- Sutton, S. M., & Rouvellou, I. (2004). Concern Modeling for Aspect-Oriented Software Development. En *Aspect-Oriented Software Development*. Addison-Wesley. (págs. 479-505).

- Suzuki, J., & Yamamoto, Y. (1999). *Extending UML with Aspects: Aspect Support in the Design Phase*. ECOOP Workshops.
- Tekinerdogan, B. (2004). ASAAM: Aspectual software architecture analysis method. *presented at WICSA 4th Working IEEE/IFIP Conference on Software Architecture*.
- Tekinerdogan, B., Hofmann, C., & Aksit, M. (2007). Modeling traceability of concerns in architectural views. En *Proc. 10th international Workshop on Aspect-Oriented Modeling AOM*.
- Tekinerdogan, B., Moreira, A., Araújo, J., & Clements, P. (2004). *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*. Workshop Proceedings.
- UML-Infrastructure. (2006). *Unified Modeling Language: Infrastructure. version 2.0*. formal/05-07-05.
- UML-Superstructure. (2005). *Unified Modeling Language: Superstructure. V 2.0*. formal/05-07-04.
- Vanhooff, B., Baelen, S. V., Joosen, W., & Berbers, Y. (2007). *Traceability as Input for Model Transformations*. Nuremberg, Germany: ECMDA Traceability Workshop.
- Walderhaug, S., Johansen, U., Stav, E., & Agedal, J. (2006). *Towards a Generic Solution for Traceability in MDD*. ECMDA Traceability Workshop (ECMDA-TW) Proceedings.
- Yu, Y., Leite, J. S. C. D. P., & Mylopoulos, J. (2004). *From Goals to Aspects: Discovering Aspects from Requirements Goal Models*. Proceedings of the Requirements Engineering Conference, 12th IEEE International. Págs.: 38-47.
- Zapata, C. M., Gelbukh, A. F., & Arango, F. (2006). Pre-conceptual Schema: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation. En *MICAI 2006*, Springer. Págs. 27-37.

# Apéndice A. Cuestionario: El tratamiento de la trazabilidad en la industria del Software.

La encuesta definida en la Tabla 35 se describen un conjunto de preguntas que facilitan indagar acerca del estado de la práctica de la trazabilidad en las empresas de desarrollo de software y empresas de servicio de la ciudad de Medellín (Col.). Los resultados de la encuesta se evidencian en el Capítulo 2. La encuesta se divide en cinco secciones que permiten entender el nivel de conocimiento que las empresas tienen de esta práctica de forma segmentada para aplicar criterios de análisis más objetivos.

**Tabla 35. Definición del cuestionario “El tratamiento de la Trazabilidad en la Industria del Software” aplicado a diferentes empresas de desarrollo de software y de servicios.**

<b>Sección A. Conocimientos Generales Acerca De La Trazabilidad</b>
1. Enuncie tres de los principales objetivos que se logran cuando se adopta una técnica de trazabilidad como parte vital para la validación y verificación de los requisitos en el ciclo de vida de desarrollo.
2. Enuncie 3 actividades que realiza para verificar la dependencia entre los requisitos?
3. Qué actividades realiza para localizar y verificar la evolución de un requisito en cualquier fase del ciclo de vida?
4. Qué nivel de experiencia tiene usted realizando actividades de trazabilidad? (Alto / Medio/ Ninguno)
5. Conoce algún método o técnica para predecir o estimar cambios en los requisitos? (Si/No -Cuál)
6. La práctica de la trazabilidad depende de complejidad de la aplicación a desarrollar? (Si/No – Por qué)
<b>Sección B. Conocimientos Específicos Acerca De La Trazabilidad</b>
1. Cuáles son las características que facilitan la ejecución de la práctica de la trazabilidad?
2. Cuáles son las características que impiden la ejecución de la práctica de la trazabilidad?
3. Qué objetivos son posibles cumplir cuando los requisitos son trazados a lo largo del ciclo de vida desarrollo?
4. Qué tipo de información se captura cuando se hace el trazado de los requisitos?
5. Cómo se captura esa información y en qué etapas del ciclo de vida?
6. Cuáles son los costos asociados a la captura de esta información?
7. Cuáles son los beneficios asociados a la captura de esta información?
8. Qué valores agregados puede representa la practica de la trazabilidad durante el proceso de desarrollo?
9. Qué métricas desde la trazabilidad son usadas en la empresa? Estas métricas corresponden a algún modelo de calidad? Cual?

<b>Sección C. Conocimientos Acerca De Modelos y Técnicas Para La Práctica</b>			
2. Nombre del modelo o técnica utilizada para la práctica de la trazabilidad			
3. Qué características considera necesarias para adoptar y usar un mecanismo o método de trazabilidad?			
4. Qué tipo de documentos de trazabilidad (físicos o electrónicos) son usados y qué vínculos o relaciones de trazabilidad pueden ser identificados en estos documentos?			
5. Cómo son creadas/ definidas y priorizadas las relaciones de trazabilidad?			
6. Cómo son los participantes involucrados en el manejo de estos vínculos?			
7. Cómo son mantenidos, revisados, actualizados o cambiados los vínculos de trazabilidad, por quién y durante qué actividad?			
8. Cuáles son los mecanismos para definir y mantener la semántica de estos vínculos?			
9. Cuál es el uso de cada tipo de vínculo de trazabilidad identificado?			
10. Utiliza alguna técnica específica para validar o verificar consistencia en los modelos que representan los requisitos en diferentes niveles de abstracción?			
11. Puede la práctica de la trazabilidad ayudar a obtener modelos consistentes? (Si/No por qué)			
12. Utiliza alguna técnica específica para validar o verificar completitud en los modelos que representan los requisitos en diferentes niveles de abstracción?			
13. Puede la práctica de la trazabilidad ayudar a obtener completitud en los modelos? (Si/No por qué)			
<b>Sección D. Trazabilidad de requisitos no funcionales y asuntos transversales</b>			
1. En el modelo de desarrollo que utiliza su empresa hacen diferencia entre los requisitos funcionales y no funcionales u otros tipos de requisitos? Cuáles? Desde qué fase del ciclo de vida se distinguen los requisitos no funcionales?			
2. Son involucrados los requisitos no funcionales en las actividades de trazado? Desde qué fase del ciclo de vida?			
3. El método utilizado para el trazado le permite realizar trazabilidad a los intereses transversales? (Si/No – Por qué)			
<b>Sección E. Herramientas Case Usadas para el Trazado</b>			
<b>Nombre de la Herramienta</b>	<b>Tipo (Comercial / GNU / Dlo Propietario)</b>	<b>Utilidades para el trazado de los requisitos</b>	<b>Fases del ciclo de vida que involucra el trazado</b>
<b>1.</b>			
<b>2.</b>			
<b>3.</b>			

# Apéndice B. Definición del Patrón de Trazabilidad en el marco de KERMETA

---

La Figura 111 se describe en la herramienta KERMET el Patrón de Trazabilidad definido en esta disertación.

```
@uri "http://traceDC"
package Traceability Pattern;
require "kermeta"

abstract class TracingModel-Engine
{
    attribute predecessor : set Element[0..1000]
    attribute axisTracing : set Element[0..1000]
    attribute successor : set Element[0..1000]
    attribute tracingLink : set TracingLink[0..1000]
    reference typeTracingModel :
kermeta::language::behavior::Literal

    operation getTypeTracingModel() : Void is abstract
    operation getAxisTracing() : Void is abstract
    operation generatePredecessor() : Void is abstract
    operation generateSuccesor() : Void is abstract
    operation getTracingLinks() : Void is abstract
}

abstract class Element inherits
kermeta::language::structure::ClassDefinition
{
    attribute composedOf : set Element [0..1000]
    reference Model : Void
    reference abstractionLevel : Void

    operation getAbstractionLevel() : set Void[0..0] is abstract
}

abstract class TracingLink
{
    reference nameLink : Void
    attribute transformationRule : set
ModelTransformations[0..1000]
    attribute targetTrace : set Element[0..1000]
    attribute sourceTrace : set Element[0..1000]

    operation getSourceTrace() : Void is abstract
    operation getTargetTrace() : Void is abstract
    operation retriveTracingLink() : Void is abstract
}
```

```

abstract class TraceSequence
{
    reference idSequence : Void
    reference sequenceInitial : Void
    reference sequenceEnd : Void
    attribute tracingModel : set TracingModel-Engine

    operation getIdSequence() : Void is abstract
    operation getTraceSequence() : Void is abstract
}

abstract class ModelTransformations
{
    attribute targetElement : set Element[0..1000]
    attribute sourceElement : set Element[0..1000]
    reference idRule : Void
    reference type : Void
    reference name : Void
    reference condition : Void
    reference subordinateOf : set TransformationRule[0..100]

    operation getRule() : set Void[0..0] is abstract
    operation transform() : set Void[0..0] is abstract
}

```

Figura 111. Descripción del modelo del Patrón de Trazabilidad definido en la herramienta KERMETA.

# Apéndice C. Definición del Casos de Estudio en Inglés: Auction System Case Study

---

El caso de estudio es descrito y desarrollado en el idioma inglés para mayor facilidad de entendimiento con otros trabajos relacionados que usan este mismo caso de estudio.

## *Problem Description*<sup>30</sup>

*The online auction system allows people to negotiate over the buying and selling of goods in the form of English-style auctions. The company owners want to rival the Internet auctioning sites, such as, eBay (www.ebay.com), and uBid (www.ubid.com). The innovation with this system is that it guarantees bid placed are solvent, making for a more serious marketplace.*

*The system only allows for enrolled users to log into the system to start a session in which they can buy, sell or browse through auctions available. Customers have credits that are used as security on each bid. Customers can increase their credit by asking the system to debit a certain amount from their credit card.*

*A customer that wishes to sell initiates an auction by informing the system of the goods to auction with the minimum bid price, the reserve price, the start period and the duration of the auction. The seller has the right to cancel the auction as long as the auction's start date has not been passed, i.e., the auction has not started already.*

*Customers of the system can search for specific items currently being auctioned. The system provides support for allowing tailored search queries, e.g. the user can specify what price he/she is willing to pay, the demanded time till auction completion, the preferred location of the seller, etc.*

*When someone wants to follow an auction, he/she must first join the auction. Once someone joined an auction, he/she may make a bid or post a message on the auction's bulletin board. A bid only is valid if it is over the minimum bid increment (calculated purely on the amount of the previous high bid, e.g., 50 cent increments when bid is between \$1-10, \$1 increment between \$10-50, etc.), and if the bidder has sufficient funds, i.e. the customer's credits within the system are at least as high as*

---

<sup>30</sup> An addaptation of [Sanen et al., 2006] Sanen, F., Truyen, E., Win, B. De., Joosen, W., Loughran, N., Coulson, G., Rashid, A., Nedos, A., Jackson, A Clarke. S. "Study on interaction issues". AOSD-Europe-ULANC-7, AOSD-EUROPE network of excellence. Feb 2006.

*the sum of all pending bids. Everyone is allowed to place one or more bids before the auction of interest closes. After the auction is closed, the system calculates the highest bid and checks if it meets the reserve price issued by the seller. If so, the system deposits the highest bid to the seller's credits and receives its commission automatically. Finally, the relevant parties are informed of the outcome of the auction.*

*At all times the system is aware of the user's context, tracking the user's current device specifications, network properties, as well as being aware of any complementary services provided by networks that are available to the user. The system is able to tailor the user's view of the online auction service to that which best suits the user's current capabilities, e.g. having the ability to reduce the volume of data being pushed at the user based on the user's screen size, storage capacity, available bandwidth etc.*

*The system provides support for client roaming in all of its forms: from clients physically changing machines to clients roaming across networks and changing IP addresses. Security is also provided to prevent spoofing attacks. Additionally the system is able to handle abrupt disconnections whereby the user cannot complete a correct logging off procedure due to network failure etc., handle any rollback issues that might arise from any incomplete transactions etc. As users may be connecting through unsecured or compromised networks the system provides its own security.*

*Users have the option to get informed of all events related to auctions in which they are involved, regardless of the fact that the user is online or offline. To realize this, the system is aware of a set of user defined rules that explain how the user wishes to interact with the system.*

Separación de Asuntos. La Tabla 36 muestra una posible separación de asuntos o subsistemas con el fin de facilitar la trazabilidad de los asuntos y los requisitos.

**Tabla 36. Separación de Interese/Asuntos/Objetivos para el Auction System.**

#	Concerns/Objectives	Requirements
1	Enrol User	1. All potential users of the system must first <i>register</i> in the system; it is a basic function to log on to the system for each session to raise.
2	Authentication	2. The buyer or seller has to log on to the system to enter in each session (it is required to make any think in the system before the enrolled).
3	Buy Goods	3. Select an active auction 4. Once a customer has joined the auction, he/she may make a bid ( <i>make bid</i> )

		<p>5. The clients bidding in parallel (bidding against each other)</p> <p>6. The client placing bids at different auctions</p> <p>7. The client increasing his/her credit in parallel</p> <p>8. Buyers can place bids until the auction closes</p>
4	Sell Goods	<p>9. A customer initiates an auction registering goods to auction with the minimum bid price and reserve price for the goods, the start period of the auction, and the duration of the auction</p> <p>10. Once an auction closes, is selected the winning bid since the system calculates whether the highest bid meets the reserve price given by the seller (English-style auction reserve price), and if so, the system deposits the highest bid price minus the commission taken for the auction service into the credit of the seller (credit internal with the system)</p> <p>11. The relevant parties are informed of the outcome of the auction</p> <p>12. The seller has the right to cancel the auction as long as the auction's start date has not been passed, i.e., the auction has not already started</p>
5	Browse auctions	<p>13. The customers are able to browse the auctions available on the system</p> <p>14. Post a message. Once a customer has joined the auction, he/she may make post a message on the auction's bulletin board (visible to the seller and all customers who are currently participants in the auction)</p>
6	Account Credit Management	<p>15. Customers have credit with the system that is used as security on each and every bid.</p> <p>16. Customers can increase their credit by asking the system to debit a certain amount from their credit card</p> <p>17. If customers want to close their auction system accounts, they can ask the system to transfer their credits back to their credit card</p> <p>18. The system must guarantee that the bid placed are solvent</p>
7	Validate bid	<p>19. A bid is valid if it is over the minimum bid increment (calculated purely on the amount of the previous high bid, e.g., 50 cent increments when bid is between \$1-10, \$1 increment between \$10-50, etc.), and if the bidder has sufficient funds, i.e. the customer's credit with the system is at least as high as the sum of all pending bids</p>
8	Performance	<p>20. The auction system is highly concurrent - clients bidding against each other in parallel.</p> <p>21. A client placing bids at different auctions and increasing his/her credit in parallel</p>
9	<p>Security:</p> <ul style="list-style-type: none"> <li>- Authentication</li> <li>- Authorization</li> <li>- Non-repudiation</li> </ul>	<p>22. The system provides support for client roaming in all of its forms: from clients physically changing machines to clients roaming across networks and changing IP addresses.</p> <p>23. Security is also provided to prevent spoofing attacks</p>

	<ul style="list-style-type: none"> <li>- Integrity</li> <li>- Confidentiality</li> <li>- Auditing</li> </ul>	<p>24. The system is able to handle abrupt disconnections whereby the user cannot complete a correct logging off procedure due to network failure etc.</p> <p>25. As users may be connecting through unsecured or compromised networks the system provides its own security.</p>
10	<p>Persistency:</p> <ul style="list-style-type: none"> <li>- State-encoding</li> <li>- State-change detection</li> <li>- State-access</li> <li>- Transactions</li> <li>- Caching</li> <li>- Logging</li> </ul>	<p>26. The system provides support for allowing tailored search queries, e.g. the user can specify what price he/she is willing to pay, the demanded time till auction completion, the preferred location of the seller, etc.</p> <p>27. The system is able to handle any rollback issues that might arise from any incomplete transactions etc.</p>
11	<p>Usability</p> <ul style="list-style-type: none"> <li>- Context-monitoring</li> <li>- Context-inference</li> <li>- Context-action</li> <li>- Profile</li> </ul>	<p>28. At all times the system is aware of the user's context, tracking the user's current device specifications, network properties, as well as being aware of any complementary services provided by networks that are available to the user.</p> <p>29. The system is able to tailor the user's view of the online auction service to that which best suits the user's current capabilities, e.g. having the ability to reduce the volume of data being pushed at the user based on the user's screen size, storage capacity, available bandwidth etc</p>