Un Patrón de Trazabilidad para Controlar la Evolución de los Intereses en un Espacio Multidimensional

Marta Silvia Tabares Betancur

Una disertación Presentada a la Facultad de Minas de la Universidad Nacional de Colombia en La Escuela de Sistemas e Informática, para la realización parcial de los requisitos para el grado de Doctor en Filosofía de Ingeniería – Sistemas

Medellín, Colombia
2009

A Traceability Pattern to Control the Concern Evolution in a Multidimensional Space

Marta Silvia Tabares Betancur

A dissertation is presented to the Minas Faculty of the National University of Colombia, in the Systems and Informatics School as part of the requirements to obtain a Doctor of Philosophy degree in Systems Engineering

Medellín, Colombia
2009

Estudiante de Doctorado:
Marta Silvia Tabares Betancur

Supervisor:
Dr. Fernando Arango Isaza (FM/Escuela de Sistemas e Informática – Universidad Nacional de Colombia, Colombia)

Co-Supervisor:
Dr. Ana Maria Dinis Moreira (CITI/Departamento de Informática, FCT/Universidade Nova de Lisboa, Portugal)

Directo Colaborador:
Dr. Raquel Anaya Hernandez (Escuela de Ingeniería, Universidad EAFIT, Colombia)

Comité de Disertación:

Dr. Jaelson Brelaz de Castro (Professor Adjunto CIn / UFPE Universidade Federal de Pernambuco. Centro de Ciências Exatas e da Natureza. Departamento de Informática).

Dr. Hernán Darío Álvarez Zapata (FM/Escuela de Procesos y Energía – Universidad Nacional de Colombia, Colombia).

Dr. João Araújo (CITI/Departamento de Informática, FCT/Universidade Nova de Lisboa, Portugal).

Reconocimientos

i. Personal

En este camino que he recorrido Dios me regaló todo lo espiritual y material necesario para lograr esta meta. En especial me regaló unas personas sabias y llenas de amor para compartir su conocimiento, experiencia y acompañarme en este camino.

En primera instancia agradezco la acogida de mi director de tesis el Dr. Fernando Arango por aceptarme como su estudiante. Él siempre tuvo una actitud positiva hacia los los avances que iba logrando. Gracias por su guía y los espacios de discusión que abrió para que avanzara en mi trabajo.

Gracias a la Dra. Ana Moreira. Persona increíble y siempre llena de amor. Me acogió en su casa, en su universidad, y me hizo parte de su grupo de investigación. Ella me dio la oportunidad de formarme como investigadora, de conocer el mundo de la investigación y la forma de actuar en este. Gracias a ella aprendí a interactuar con otros investigadores de diferentes países, aprendí a escribir y evaluar artículos. También me dio la oportunidad de acompañarla como colega en sus clases en la Universidade Nova de Lisboa. Gracias a su hermosa familia por adoptarme por un tiempo y hacerme partícipe de algunas de sus cosas en mi estadía en Portugal. Espero poder seguir trabajando con ella en labores académicas y de investigación, siempre será un honor para mí poder hacerlo.

Gracias a la Dra. Raquel Anaya, mi gran maestra e imagen de profesional que me inspiró. Ella me abrió el camino como investigadora y me facilitó los medios para dar a conocer mis trabajos, en diferentes paises. Me acompañó y guió en las labores de investigación, puso a disposición su grupo de investigación para que yo lograra mis metas, y me dio su compañía y amistad para salir adelante en esta difícil tarea. A ella mil gracias y espero poder seguir trabajando con ella en labores académicas y de investigación, siempre será un honor para mí poder hacerlo.

Gracias a los Jurados porque me dieron desde diferentes instancias la posibilidad de compartir experiencias académicas y de investigación con ellos. Además, por sus correcciones y sugerencias que me ayudaron a lograr un documento final de calidad.

Gracias a mis amigos. A Jonh Willam Branch que me motivó a explorar este perfil profesional de mi carrera y me acompaño durante este tiempo con su energía positiva para que yo tuviera

confianza en lo que podía lograr. A Carlos Mario Zapata, mi compañero de estudio que siempre con su gran corazón me impulsó y valoró el trabajo que realizaba. A Marta Cecilia Meza que siempre confió en mí como su compañera de trabajo y amiga proporcionándome lo que estuviera a su alcance para que yo lograra las metas de investigación que fortalecieron este logro. A mi hermana del alma Mónica Mezquita, PhD en Educación, que vive en Portugal y que con su familia me acogió proporcionándome un hogar para siempre. A mis amigos Jorge Iván Gómez y Luis Norberto Parra que me dieron su confianza y apoyo desde que tuve la idea de hacer el doctorado.

También agradezco a mis estudiantes de la Escuela de Ingeniería de Antioquia, otros docentes y amigos que me apoyaron directa o indirectamente, y estuvieron al pendiente de mis avances y que de alguna u otra forma siempre me apoyaron.

Finalmente, y lo más importante, quiero dar gracias a mis amados Padres Luz Socorro Betancur e Ignacio Tabares, a mis abuelos Herminia Ochoa y Carlos Betancur (ya fallecidos), a mi hermano Carlos Ignacio y mis hermanas Claudia Maria y Natalia, que con su infinito apoyo y amor siempre confiaron que yo lograría esta meta.

ii. Institucional

Agradezco a la Universidad Nacional de Colombia – Sede Medellín por su apoyo en el inicio de mi doctorado cuando aún me desempeñaba como la directora del Centro de Informática de esta gran institución. En especial quiero agradecer al profesor Dr. Oscar Mesa, él con su conocimiento, disposición y sabias palabras confió desde el principio que yo podría lograr lo que me propusiera.

Agradezco a la Escuela de Ingeniería de Antioquia. Me acogió como parte de su grupo de trabajo y me proporcionó todos los recursos para desarrollar mis estudios y realizar mis viajes a congresos y otros eventos de carácter investigativo.

iii. Soporte Financiero

Agradezco a las siguientes instituciones por el apoyo financiero que me brindaron para la realización del doctorado y otras actividades asociadas a este.

- Colciencias (Instituto Colombiano para la Educación y la Investigación) que me otorgó una beca condonable para realizar mis estudios de doctorado.
- Universidad Nova de Lisboa que me apoyó con una beca durante mi estadía en el segundo semestre de mi pasantía.

A mí amada familia, fortaleza inconmensurable en la formación de mi ser

Yo no soy dueña de nada, ni del conocimiento, ni de mi misma. Solo soy responsable de mi cuerpo, mi mente, mi espíritu, y de lo que haga con ellos. Por eso ahora, con esta obra intelectual de la cual me hice responsable, se crea un estilo, un que hacer, un pensamiento, que espero trascienda para el beneficio del conocimiento de quienes lo utilizan.

Resumen

La trazabilidad de requisitos es un atributo de calidad tratado por la ingeniería de software y en especial por la ingeniería de requisitos. Hace parte de las prácticas de la gestión de la configuración del software o la gestión del cambio que particularmente hace el seguimiento al rastro (hacia delante y hacia atrás) que dejan los requisitos y los artefactos que los representan cuando estos sufren cambios durante el proceso de desarrollo del software o durante su mantenimiento.

Además es una práctica que ayuda a las tareas de validación y verificación que se aplica una y otra vez con una misma técnica para rastrear la evolución de los requisitos desde su definición hasta convertirse en productos de software.

Rastrear los requisitos es una actividad necesaria para mantener el control de calidad y exactitud de los productos de software requeridos por el cliente. No obstante, la práctica de la trazabilidad es vulnerable a las prácticas de desarrollo de software y de calidad que los grupos de adopten durante el proceso de desarrollo de software. También está sujeta a otros factores que podrían entorpecer su constante práctica en la industria del software: complejidad de los modelos, carencia del registro de la evolución de los requisitos, cambios invasivos no controlados (enredados o dispersados), modelos parcialmente actualizados cuando hay cambios, matrices de trazabilidad no actualizadas, altos costo para la realización de esta práctica, entre otros. Además, los diferentes enfoques de trazabilidad que se han propuesto no son fáciles de adoptar para los modelos de desarrollo actuales.

Aunque varios enfoques para manejar la trazabilidad han sido propuestos, esta práctica sigue siendo un tema de investigación interesante, principalmente porque el desarrollo de software es una disciplina en evolución constante. Nuevas técnicas, herramientas, metodologías y paradigmas siguen apareciendo y exigiendo mejoras constantes a las prácticas de la gestión de la configuración del software para garantizar productos mejor calificados y menos costosos.

Esta disertación define un nuevo enfoque de trazabilidad en el contexto de dos paradigmas de desarrollo de software emergentes, el desarrollo de software dirigido por modelo MDD (*Model-Driven Development*) y el desarrollo de software orientado por aspectos AOSD (*Aspect-Oriented Software Development*) los cuales proporcionan estrategias de desarrollo y mecanismos que ayudan a definir un enfoque para minimizar algunos de los problemas antes presentados. Mientras MDD contribuye con patrones que facilitan y estandarizan la transformación de modelos en diferentes

niveles de abstracción, AOSD proporciona modos de separar asuntos y componer asuntos transversales.

Esta disertación trata la trazabilidad como un patrón que controla la transformación de los modelos en el contexto del desarrollo de software dirigido por modelos, y específicamente los modelos de asuntos. Esto garantiza características de calidad como la modificabilidad, la consistencia, la completitud y la propagación del cambio. Para lograr esto, se define un metamodelo denominado "Patrón de Trazabilidad" que soporta la creación y gestión de Modelos de Trazado, y de Modelos de Asuntos.

Los Modelos de Trazado son elementos abstractos compuestos por elementos trazables y vínculos de trazado para facilitar su uso en diferentes proyectos de desarrollo. Por medio de estos modelos se establece un marco conceptual y metodológico de trazabilidad para lograr las características de calidad antes mencionadas y apoyar de forma ágil a los equipos de desarrollo durante el proceso de desarrollo y en las tareas de mantenimiento del software. Algunas de estas características son:

- Estandarización de los vínculos de trazado definidos por UML para controlar la transformación de modelos.
- Control de predecesores y sucesores generados durante el proceso de transformación para facilitar la consistencia y completitud de los modelos de desarrollo.
- Generación de vistas versionadas de Modelos de Trazado para realizar la gestión del cambio.

Por otro lado, los Modelos de Asunto son elementos abstractos que definen la separación de asuntos y las posibles composiciones entre estos. En otras palabras, agrupan asuntos de diferente tipo que se interelacionan entre si por relaciones que expresan el tipo de dependencia o composición. Cada instancia de modelo representa un subsistema u objetivo de un sistema que puede ser trazado y transformado como un todo. Los Modelos de Asuntos se definen como un enfoque que aplica a la ingeniería de requisitos orientada a aspectos cuya trazabilidad y transformación se soporta en los Modelos de Trazado.

Abstract

Requirement traceability is a quality attribute handled by software engineering and especially by requirement engineering. This is an activity that is needed in order to support the quality control and the accuracy of the software products in accordance with the client's needs.

The traceability is part of the software configuration management (change management) practices. Particularly, this one follows the requirement tracks (forward and backward) either during the software process or by changes suffered during its maintenance. Furthermore, the traceability is a practice that helps with both validation and verification tasks. This practice is applied repeatedly using the same technique to trace the requirement evolution and transformation from its definition to its becoming a software product.

Nevertheless, the traceability practice is vulnerable to the software development process and quality practices that are adopted by the development team during the process. There are other factors that might obstruct the practice in the software industry: the model complexity, lack of the requirement evolution record, invasive changes that are not controlled (tangled or scattered), partially updated models when there are changes, outdated traceability matrices, and high costs to perform this practice, etc. Moreover, some traceability approaches that have been proposed are not easy to be adapted by the current development processes.

Although several traceability approaches have been proposed, this is still a topic under investigation. This occurs mainly because software development is a discipline in constant evolution.

New techniques, tools, methodologies, and paradigms continue to appear and to demanding constant progress from the software configuration management practices to guarantee better qualified and less costly software. Therefore, this dissertation defines a new traceability approach in the context of two emergent software development paradigms, the model-driven software development MDD, and the aspect-oriented software development AOSD, both which provide strategies of development and mechanisms that help to define an approach to minimize some of the problems presented earlier. While the MDD contributes with patterns to facilitate and standardize the model transformations in different levels of abstraction, AOSD provides ways to separate concerns and compose crosscutting concerns.

This dissertation treats the traceability as a pattern that controls the model transformations in the context of the model-driven development, and specially the concern model transformation. This guarantees quality characteristics such as modifiability, consistency and completeness, and change propagation. To achieve this, we define a metamodel called “Traceability Pattern” that supports the creation and management of Tracing Models, and Concerns Models.

A Tracing Model is an abstract element composed of traceable elements and tracing links to facilitate its use in different software development projects. By means of these models, there is a conceptual and methodological traceability framework established in order to achieve the quality characteristics mentioned earlier. Also, these agilely support the development team in the software maintenance tasks. The main features that treat the Tracing Models are:

- Standardization of the UML dependency relationships as tracing links to control the transformation of models.
- Control of predecessors and successors generated during the transformation process, in order to facilitate both consistency and completeness verification of development models.
- Generation of versioning views of Tracing Models to achieve change management.

On the other hand, the Concern Model is an abstract element that defines the separation of concerns and their possible compositions. They group concerns of different types related among themselves by relationships that define the type of dependency or composition. Every instance of model represents a subsystem that can be traced and transformed as a whole. Concern Models are defined as an aspect-oriented requirement engineering approach whose traceability and transformation is supported by the Tracing Models.

Acrónimos

AORA: Aspect-Oriented Requirements Analysis

AORE: Aspect-Oriented Requirement Engineering

AOSD: Aspect-Oriented Software Development

ARCaDE: Aspectual Requirements Composition and Decision SupportATL: ATLAS Transformation Language

CORE: Concern Oriented Requirements Engineering

MDA: Model-Driven Architecture

MDD: Model-Driven Development

MOF: Metamodel Object Facility

NFR: Non-Functional Requirements

OMG: Object Management Group

QVT: Query-View-Transformation

SCM: Software Configuration Management

UML: Unified Modeling Languages

XML: Extensible Markup Language.

Glosario

- Artefacto de Software: cualquier pieza de software usada durante el desarrollo y el mantenimiento del software. Describe la función, la arquitectura, y el diseño del software. Algunos ejemplos de artefactos son: casos de uso, diagramas de clases, documentos de requisitos, y del diseño. Otros artefactos están relacionados al proceso del negocio, por ejemplo, un plan de proyecto, casos del negocio, evaluación del riesgo (Wikipedia 2008).
- Completitud (*Completeness*): una especificación es completa al grado que todas sus partes están presentes y cada parte es totalmente desarrollada. Una especificación de software debe exponer varias propiedades para asegurar su completitud. [Boehm 1984].
- Asunto (*Concern*): objetivo o **interés** del sistema que puede ser identificado, modelado y desarrollado como un producto de software. Este puede ser desde un requisito, un modelo, un artefacto de software, entre otros elementos que representen un interés del sistema.
- Consistencia (*Consistency*): una especificación es consecuente al grado que sus provisiones no entran en conflicto el uno con el otro o con especificaciones gobernantes y objetivos. Las especificaciones requieren consistencia de diferentes formas [Boehm 1984].
- Asunto Transversale (*Crosscutting Concern*): elemento de modelo identificado en las etapas tempranas del desarrollo que pueden cruzar el comportamiento de otros asuntos del sistema.
- Elemento de modelo (*Model Element*): es un elemento que puede ser representado en un lenguaje de modelado. UML define cada uno de sus elementos como un elemento de modelo.
- Enfoque (*Approach*): es una propuesta con un fin específico. En esta disertación se usa la palabra como la proyección o la aproximación de la idea o el objetivo desarrollado en cada capítulo.
- Espacio Multidimensional: abstracción del espacio con más de tres dimensiones, a diferencia del espacio habitual (estudiado en la geometría elemental). En el espacio multidimensional de n dimensiones, la posición de un punto se caracteriza mediante n números (con la particularidad de que el espacio puede tener un número finito e infinito de dimensiones) [Diccionario Filosófico 1965].
- Marco de Trabajo (*Framework*): es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado (Definición tomada de Wikipedia).

- Modelo: un modelo corresponde a un conjunto coherente de elementos que representan una situación determinada, y es interpretado de acuerdo a una forma de análisis específica [Mellor et al. 2003]. Por lo tanto, es importante mantener su significado y el de sus elementos con respecto a los requisitos, por medio de la verificación de completitud y consistencia de sus elementos a través del ciclo de vida de desarrollo.
- Modificabilidad (*Modifiability*): se refiere a la capacidad de modificación; estado o calidad de ser modificable [Bachmann et al. 2007].
- Stakeholder: persona, grupo, organización, o sistema quién afecta o puede ser afectado por el éxito o el fracaso de un proyecto de desarrollo de software. Ellos podrían ser el Patrocinador, Visionario, Embajador, Usuario, u otros actor según la metodología usada durante el proceso de desarrollo.
- Traza (*Trace*):
  - Seguimiento, descubrimiento, o averiguación del curso de desarrollo de algo. [Word Magic-Translation Software. http://www.wordmagicsoft.com/diccionario/en-es/trace.php (cited: 10/8/2007)].
  - Una señal visible, como una huella, hecha o dejada por el paso de una persona, animal, o cosa.
  - Pruebas o una indicación de la antigua presencia o existencia de algo; un vestigio. [Answers.com http://www.answers.com/trace (cited 10/8/2007)].
- Trazado (*Tracing*):
  - El trabajo de investigación de algo a fondo y sistemáticamente [Word Magic-Translation Software. http://www.wordmagicsoft.com/ diccionario/en-es/tracing.php (cited: 10/8/2007)].
  - El trazado de software provee a los desarrolladores información útil para la depuración. Esta información es usada tanto durante el ciclo de desarrollo como después de que el software es liberado [Answers.com http://www.answers.com/trace (cited 10/8/2007)].
- Trazabilidad (*Traceability*): trazabilidad o rastreabilidad se refiere a la completitud de la información sobre cada paso en una cadena de proceso. Definición básica: la trazabilidad es la capacidad de interrelacionar por orden cronológico las entidades únicamente identificables en un camino que es verificable. La trazabilidad es la capacidad de verificar la historia, la posición, o la aplicación de un artículo por medio de la identificación registrada y documentada

[Wikipedia The Free Encyclopedia. http://en.wikipedia.org/wiki/Traceability (citada: 10/8/2007)].

- Transformación (*Transformation*): proceso de convertir a un modelo Ma conforme a un metamodelo MMA en un modelo Mb conforme a un metamodelo MMb. Si MMa=MMb, entonces uno tiene una transformación endogena; de otra forma otro tiene una transformación exogena. La transformación de modelos usualmente toma lugar usando lenguajes de transformación tales como GReAT, VIATRA, ATL, etc. [Wikipedia The Free Encyclopedia. http://en.wikipedia.org/wiki/Traceability (cited: 10/8/2007)].
- Validación (*Validation*): proceso de evaluar el software al final de proceso de desarrollo de software para asegurar la conformidad con los requisitos de software [Boehm 1984]. La validación se refiere al proceso de evaluar el software durante el proceso de desarrollo o en el final, a fin de determinar si este satisface los requisitos especificados. Este es realizado por los stakeholders para asegurar que el producto reúne las necesidades del usuario y que las especificaciones fueron correctas [IEEE-STD-610].
- Verificación (*Verification*): proceso para determinar si los productos de una fase dada del ciclo de desarrollo de software realizan las exigencias establecidas durante la fase anterior [IEEE-STD-610].

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1
Introducción

La evolución de los requisitos es inevitable durante el proceso de desarrollo de software. Está relacionada con factores tales como la variabilidad de reglas comerciales y cambios en las decisiones tecnológicas que apoyan la arquitectura del sistema. Por lo tanto, los modelos de software deberían conseguir un grado bueno de modificabilidad y trazabilidad con el fin de atender a atributos de calidad como estabilidad, complejidad, completitud, y consistencia, entre otros.

A menudo, la práctica de la trazabilidad se realiza durante la ingeniería de requisitos, actividades de pruebas (*testing*), o el mantenimiento de aplicaciones. La trazabilidad se soporta en matrices tales como Requisitos vs. Casos de Uso, o Casos de Uso vs. Prototipos, etc. Sin embargo, éstas no se actualizan con frecuencia mientras el proceso de desarrollo avanza o la mayor parte de los cambios son hechos en elementos modelos en etapas de desarrollo avanzadas que no son controlados por esas matrices.

Así, cuando un cambio es hecho, los desarrolladores no tienen el conocimiento suficiente para saber qué elementos de modelo son afectados en diferentes niveles de abstracción. Esto afecta actividades complementarias como la valoración del impacto del cambio, o la identificación de inconsistencias o incompletitudes de los modelos de desarrollo.

Varios enfoques de trazabilidad se han definido para garantizar el trazado y la correlación entre los requisitos y sus artefactos de refinamiento a través del ciclo de vida de desarrollo: [Gotel and Finkelstein 1997], [Lindvall 1994], [Ramesh and Jarve 2001], [Egyed 2003], [Cleland-Huang 2003], [Aizenbud-Reshef et al. 2005], etc. Aunque estos enfoques proporcionan semántica

enriquecida y mecanismos para hacer la trazabilidad de requisitos, a menudo no son adoptados por equipos de desarrollo de la industria del software.

Los problemas sobre el uso y adopción de la trazabilidad eran y seguirán siendo un tema de estudio. Por ejemplo, [Gotel and Finkelstein 1994] y Ramesh [Ramesh 1998] hicieron grandes investigaciones para identificar que tipo de problemas podrían afectar esta práctica. Los resultados entregados principalmente mostraron el compromiso o la participación confusa de los stakeholders y equipo de desarrollo (Gerente de proyecto, Analista, Arquitecto, Desarrollador, Probador, etc.) en las actividades de trazabilidad; de igual forma el uso de la información de trazado para validar y verificar la evolución de requisitos en sus modelos de desarrollo.

Por otra parte, estudios recientes están orientados a la identificación de características de trazado que proporcionan paradigmas de desarrollo emergentes tales como la Ingeniería dirigida por modelos y el Desarrollo de Software Orientado a Aspectos con el fin de saber como sus acercamientos pueden proporcionar el apoyo o falicitar el mecanismo para realizar la trazabilidad de los requisitos [Chitchyan et al. 2005], [Czarnecki y Helsen, 2006], [Aizenbud-Reshef et al., 2006], [Galvão y Goknil 2007], [Galvão et al. 2008].

La trazabilidad en el MDD es tratada como el modo de registrar la operacionalización de las transformaciones de los modelos con el fin de apoyar actividades como propagación de cambio, y la verificación de inconsistencias e incompletitudes [Mens and Van Gorp 2006]. Trazabilidad en el AOSD es tratada desde la separación de asuntos y la composición de asuntos transversales para controlar los defectos que pueden ocurrir por cambios invasivos durante el proceso de desarrollo. Por lo tanto, este paradigma es orientado para evitar o advertir sobre elementos enredados o dispersos en todas partes del proceso de desarrollo [Chitchyan et al. 2005], [Berg et al. 2006].

Una vez se conocen enfoques representativos en el dominio de la trazabilidad, que dedican una buena cantidad de trabajo al modelado de la trazabilidad, es importante identificar realmente cuál es el valor de la práctica de la trazabilidad para las empresas de desarrollo de software o las empresas de servicio que la practican. Antes de definir una nueva teoría para la gestión de la trazabilidad, es necesario formular algunas preguntas:

- ¿Por qué la práctica de la trazabilidad se considera importante pero no se hace de forma consistente para cualquier tipo de proyecto de software?
- ¿Los grupos de desarrollo son conscientes de cuánto cuesta No hacer trazabilidad, existe algún análisis costo-beneficio al respecto?

- ¿Por qué la práctica de la trazabilidad se realiza la mayor parte del tiempo en la fase de los requisitos, conociendo de su importancia durante todo el proceso de desarrollo?
- ¿Es posible adoptar nuevos paradigmas de desarrollo para crear soluciones más cercanas a la realidad de la práctica de la trazabilidad?

Esta disertación es definida en los contextos de MDD y AOSD para garantizar la trazabilidad de asuntos y asuntos transversales durante la transformación de modelos, al igual que se logra su consistencia y completitud durante el proceso de desarrollo. Por lo tanto, se define un patrón de trazabilidad en el nivel del metamodelo con el fin de proporcionar el mecanismo y elementos necesarios para controlar la transformación de los modelos y proporcionar servicios para soportar la gestión de la configuración. La Figura 1 muestra cuatro escenarios de la solución que se desarrollan a lo largo de esta disertación.

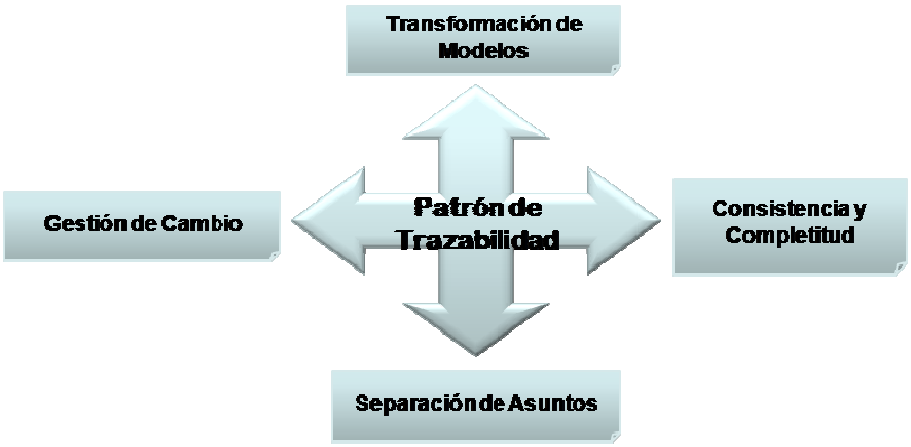

Figura 1. Escenarios de la Solución que describen esta disertación.

A continuación se enuncian los escenarios en los cuales se desarrolla el Patrón de Trazabilidad.

- **Transformación de modelos (*Model Transformations*)**. La transformación de modelos es definida por el patrón de trazabilidad. Por medio de modelos de trazado abstracto se facilitan la definición y tratamiento de la trazabilidad en cualquier ambiente de desarrollo. Estos modelos de trazado actúan como patrones que estandarizan la práctica de la trazabilidad facilitando la gestión de la configuración que soportan el proceso de desarrollo.
- **Consistencia y Completitud (*Consistency and completeness*)**. El patrón de trazabilidad proporciona el mecanismo de verificación de la consistencia y la completitud de los modelos de desarrollo que son transformados en diferentes niveles de abstracción.

- **Separación de Asuntos (*Separation of Concerns*)**: la separación de asuntos se aplica desde dos puntos de vista: (1) los elementos de modelo trazables que son la parte de un modelo de trazado son separados en tres roles con responsabilidades específicas definidas por el patrón, y (2) la definición de modelos de asuntos que facilitan la transformación y el rastro de asuntos transversales en diferentes niveles de abstracción.
- **Gestión del Cambio (*Change Management*)**: los modelos de trazado son construidos con características de modifiability para proporcionar la propagación de cambio.

La estructura de este capítulo se define a continuación. La Sección 1.1 presenta diferentes aspectos que motivaron el desarrollo de esta tesis. La Sección 1.2 define la hipótesis de la tesis. La Sección 1.3 define los objetivos de la tesis. La Sección 1.4 presenta la contribución y alcance de esta tesis. La Sección 1.5 define la estructura de la tesis. La Sección 1.6 presenta las publicaciones que soportan el desarrollo de esta tesis.

1.1 Motivación

Esta disertación responde a los siguientes cuestionamientos:

- ¿Por qué la práctica de la trazabilidad en la industria de software no se estandariza con un artefacto de control de la evolución de los requisitos con actividades de transformación implícitas durante el proceso de desarrollo?
- ¿Cómo puede la separación de asuntos, tratada por el desarrollo de software orientado por aspectos, facilitar la trazabilidad de requisitos?
- ¿Es posible definir un modelo de trazabilidad para trazar los asuntos y asuntos transversales?
- ¿Cómo puede la trazabilidad controlar la transformación de modelo y garantizar su consistencia y completitud?
- ¿Es posible medir el esfuerzo que requiere la gestión del cambio durante la transformación de modelos?
- ¿Cuál es el valor agregado que genera el tratamiento de la trazabilidad desde el contexto del desarrollo de software dirigido por modelos?

1.2 Hipótesis

La tesis que se desarrolla en esa disertación se describe a partir de las siguientes hipótesis:

- **Hipótesis 1**. La trazabilidad es una práctica de validación y verificación que se realiza iterativamente durante el proceso de desarrollo. Por lo tanto, la trazabilidad se puede tratar como un patrón.
- **Hipótesis 2**. Es posible definir y aplicar un patrón de trazabilidad para controlar la transformación de los asuntos y asuntos transversales.
- **Hipótesis 3**. Es posible usar el patrón de trazabilidad para controlar el impacto del cambio.

1.3 Objetivos

El objetivo de esta tesis es definir un enfoque que facilite la trazabilidad de asuntos y asuntos transversales durante la transformación de modelos de tal forma que se garantice su consistencia y la completitud a través de diferentes niveles de abstracción.

Objetivos Específicos

- Identificar diferentes características, elementos y comportamiento que sean necesarios para establecer un modelo de trazabilidad que soporte la evolución y transformación de los asuntos y asuntos transversales.
- 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.

1.4 Contribución y Alcance de la Tesis

La contribución de esta tesis es conceptual y metodológica. Presenta un método para tratar la trazabilidad de requisitos como un patrón que facilite la gestión de los modelos de trazado que controlan la transformación de modelos.

La contribución está orientada a:

- Hacer la revisión del estado del arte de la trazabilidad de requisitos con el fin de identificar como se logra la consistencia y la completitud de los modelos a partir de esta práctica. Un conjunto de enfoques de trazabilidad se analiza desde contextos diferentes de desarrollo. Así, sus fortalezas y debilidad son identificadas para obtener diferentes escenarios que podría atender esta disertación.
- Definir un metamodelo que soporte la creación de Modelos de Trazado, la transformación de modelos de asuntos, la verificación de la consistencia y la completitud, y la gestión del cambio.
- Definir un modelo de transformación que facilite el manejo de reglas de transformación para garantizar la evolución de requisitos y artefactos de software identificados en diferentes niveles de abstracción.
- Identificar un método para gestionar el impacto y la propagación del cambio desde la transformación de modelos.
- Establecer un marco metodológico que permita adoptar el método de trazabilidad establecido a partir de los modelos de trazado y su control de la transformación de modelos.

El alcance de esta disertación está limitado por los siguientes aspectos:

- El enfoque de trazabilidad sólo se define para ser utilizado en las etapas tempranas del proceso de desarrollo: Requisitos, Análisis, y Arquitectura de Diseño.
- El enfoque de trazabilidad no incluye la verificación de la consistencia semántica en sentencias declaradas en descripciones del problema, características, requisitos o necesidades de usuario.
- El enfoque de trazabilidad se desarrolla en el contexto de la notación UML u otras compatibles con el metamodelo MOF. De otra forma, otros tipos de notaciones no son usadas, ni tratadas en esta disertación.
- La disertación no tiene el objetivo de crear un lenguaje de transformación. Se usan expresiones declarativas para describir las reglas de transformación y se compara su declaración en otros lenguajes de transformación como ATL y QVT.
- Esta disertación no tiene el objetivo de desarrollan una herramienta de modelado que soporte el enfoque de trazabilidad.

- Este enfoque no trata la trazabilidad desde ontologías, ni hace análisis léxico de requisitos u otras descripciones para trazar los requisitos a través del proceso de desarrollo.
- La disertación valida resultados por medio de un caso de estudio de carácter académico que facilita la demostración del enfoque de trazabilidad definido. Además se evalúan métricas para transformación modelos e impacto de cambio.

1.5 Estructura de la Tesis

La estructura general de esta tesis está dividida en ocho capítulos donde cada uno define un contenido o enfoque que apoya la demostración de la tesis. Para mayor entendimiento de los lectores, en la Figura 2 se ilustra el orden lógico de los capítulos en tres secciones que los agrupan de acuerdo a su aporte en el cumplimiento de los objetivos planteados.

Para cada capítulo se propone una estructura guiada por un resumen, una introducción, el desarrollo del enfoque, la discusión (sólo cuando aplique), trabajos relacionados (sólo cuando aplique), y conclusiones.

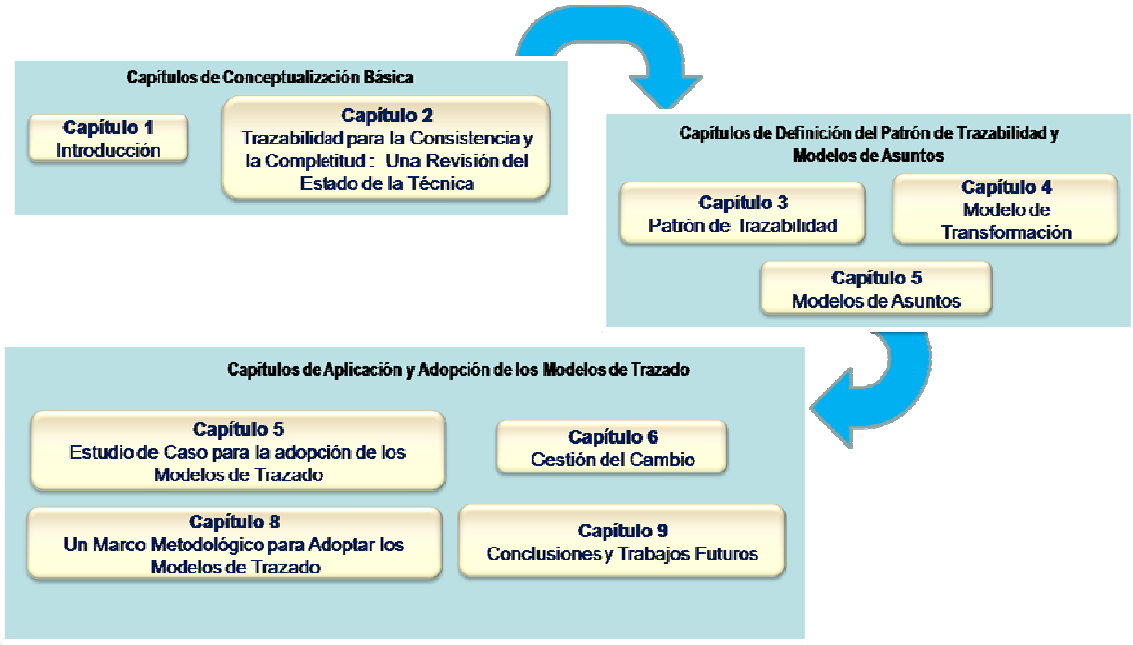

Figura 2. Estructura de la tesis.

Sección 1: Capítulos de Conceptualización Básica. Incluye los capítulos que conceptualizan e introducen el tratamiento de las teorías básicas y el estado del arte de la trazabilidad de requisitos.

- **Capítulo 1. Introducción**

La introducción es tratada en este capítulo como el conjunto de secciones que motivan la definición y esquema de la tesis. Está se complementa con siete capítulos más, las referencias bibliográficas, un apéndice que es describe brevemente en la parte final del documento elementos de tratamiento general tales como el estudio de caso, el formulario de investigación acerca del uso de la trazabilidad y otras figuras usadas en diferentes capítulos.

- **Capítulo 2. Trazabilidad para la Consistencia y la Completitud: Una revisión del estado de la técnica**

En este capítulo se hace una revisión del estado de la práctica de la trazabilidad. Para logra esto se cumple dos objetivos: (1) definir la trazabilidad, la consistencia y la completitud desde sus conceptos básicos, técnicas y características generales en su tratamiento; (2) evaluar los enfoques de trazabilidad desde diferentes contextos del desarrollo de software con el fin de identificar sus modelos, características, y la forma como logran la consistencia y la completitud de los modelos de desarrollo. Como resultado de esta revisión se destacan los aspectos más relevantes de cada enfoque y se hace una discusión de posibles fortalezas y debilidades que presentan para el tratamiento de la trazabilidad en especial lo que corresponde a la consistencia y la completitud. Esto facilita el planteamiento de los escenarios de trabajo que se desarrollan en esta disertación.

Sección 2: Capítulos de Definición del Patrón de Trazabilidad. Incluye los capítulos que describen las teorías y modelos creados para tratar la trazabilidad como un patrón en el contexto del desarrollo dirigido por modelos.

- **Capítulo 3. El Patrón de Trazabilidad**

En este capítulo se define un patrón de trazabilidad que facilita la demostración de la tesis. Este patrón estandariza la construcción y la utilización de modelos de trazado que controlan las transformaciones de los modelos y garantizan su consistencia y la completitud en diferentes niveles de abstracción. Un método de trazabilidad apoyado en el caso de estudio se define para validar la aplicación de los modelos de trazado durante el proceso de desarrollo.

- **Capítulo 4. El Modelo de Transformación**

En este capítulo se define el modelo de transformación que soporta los modelos de trazado. Este proporciona características que definen un ambiente de transformación y la operacionalización de reglas de transformación cuya ejecución es controlada por los modelos de trazado. Además, se define un planificador que establece un mecanismo para garantizar la ejecución de las reglas que

lograr el refinamiento y la correlación de los elementos modelos trazables definidos por los modelos de trazado.

- **Capítulo 5. Modelos de Asuntos**

En este capítulo se define un enfoque llamado Modelo de Asuntos. Este establece estrategias de separación de asuntos como base para definir Modelos de trazado y consecuentemente controlar su transformación en diferentes niveles de abstracción. Los modelos de asuntos son una forma de facilitar la descomposición y la composición de asuntos en diferentes niveles de abstracción. Los asuntos se identifican por tipos que guían la separación de un problema por características de funcionalidad, información, reglas del negocio, servicio de calidad, y contextos. Estos son transformados desde el espacio del problema a través de diferentes niveles de abstracción.

Sección 3: Capítulos de Aplicación y Adopción de los Modelos de Trazado. Incluye los capítulos que describen la forma de aplicar y adoptar los modelos de trazado desde el contexto de un estudio de caso, la definición de la gestión del cambio, y de un marco metodológico que soporta la trazabilidad como asunto que controla la transoformación de modelos.

- **Capítulo 6. Estudio de Caso para la adopción de los Modelos de Trazado**

En este capítulo, por medio de un estudio de caso, se describe un método de trazabilidad que soporta el uso de los modelos de trazado.

- **Capítulo 7. Gestión del Cambio**

En este capítulo se define un enfoque para gestionar el cambio desde los modelos de trazado. El enfoque identifica las características que determinan el impacto del cambio y su propagación. Además se define la métrica que ayuda a medir el esfuerzo que un equipo de desarrollo debe hacer cuando un acontecimiento de cambia los requisitos del sistema.

- **Capítulo 8. Un Marco Metodológico para Adoptar los Modelos de Trazado en el contexto de MDD**

En este capítulo se define un marco metodológico que apoya un proceso de desarrollo dirigido por modelo controlado por modelos de trazado. Este marco es un enfoque que proporciona un conjunto de actividades, recursos, tarea y entregables que son determinados la transformación de los modelos en diferentes niveles de abstracción. Así, la gestión de configuración es vista como un conjunto de servicios de trazabilidad que controlan el proceso de desarrollo. La definición de marco se hace en el Eclipse Process Framework Composer (EPFC) con el fin de validar su aplicabilidad.

- **Capítulo 9. Conclusiones y Trabajo Futuro**

En este capítulo se concluye de acuerdo a los objetivos establecidos para esta tesis. Además, se describen los resultados obtenidos por cada capítulo, y el trabajo futuro que se realizará a partir de dichos resultados.

1.6 Publicaciones de la Tesis

Las siguientes publicaciones se lograron durante el proceso de identificación, creación y aplicación de la tesis:

1. Tabares, M.S., Zapata C.M., Arango, F. (2005). "Un Método para el Refinamiento de los Atributos Derivados del Diagrama de Clases", Revista Av. Sist Inf., Vol 2 No. 1, págs: 1-8, Medellín, Junio 2005, ISSN 1657-7663.
2. Tabares, M.S. (2005). "Análisis de la trazabilidad desde la Orientación a Aspectos", Revista EIA. ISBN 1794-1237. Vol. 000. No.0004. 2005.
3. Tabares, M., Moreira, A. (2006). "Towards a Meta Aspect for Traceability". Early Aspects: Traceability of Aspects in the Early Life Cycle Workshop. AOSD'06. March 2006, Germany.
4. Tabares, M.S., Moreira, A., Arango, F., Anaya, R., Araújo, J. (2006). "Semántica de Trazado para Asuntos Transversales". Workshop DSOA'06 in JISBD 2006, Sitges (Barcelona).
5. Tabares, M.S., Arango, F. y Anaya, R. (2006). "Una revisión de modelos y semánticas para la trazabilidad de requisitos". Revista EIA (ISSN 1794-1237), número 6, Diciembre 2006, págs: 33-42.
6. Tabares, M.S., Moreira, A., Anaya R., Arango F., Araújo J. (2007). "A Traceability Method for Crosscutting Concerns with Transformation Rules". In: Proceddings IEEE Early Aspects at ICSE: Workshop in Aspect-Oriented Requirements Engineering and Architecture Design (EARLYASPECTS'07) in 29th ICSE, US. 2007. ISBN: 0-7695-2830-9.
7. Tabares, M.S. y Arango, F. (2007). "La Trazabilidad de Requisitos vista desde las Nuevas Tendencias de la Ingeniería de Software". Evento de Investigación: Tendencias en Ingeniería de Software e Intenligencia Artificial. Universidad Nacional de Colombia, Medellín. Ed. por C.M Zapata y G.L Giraldo. Colombia. 2007. ISBN: 978-958-44-1444-4.

8. Tabares, M.S., Anaya, R., Arango, F. (2007). "La Ingeniería de Requisitos Orientada a Aspectos: Una Experiencia de Aplicación en un Sistema de Ayuda en Línea". Revista DYNA No 153 Noviembre 2007. ISSN 0012-7353.
9. Tabares, M.S., Barrera, A., Arroyave, J.D, Pineda, J.D. (2007). "Un Método para la Trazabilidad de Requisitos en el Proceso de Desarrollo Unificado". Revista EIA No. 8. ISSN 1794-1237.
10. Tabares, M.S., Anaya, R., Arango, F. (2008). "Un Esquema de Modelado para Soportar la Separación y Transformación de Intereses durante la Ingeniería de Requisitos Orientada por Aspectos". III Congreso Colombiano de Computación, Medellín, Abril 23-25 de 2008. Revista Avances en Sistemas e Informática, Vol. 5, Nº. 1, 2008, págs: 189-198. ISSN: 1909-0056.
11. Tabares, M.S., Anaya R., Moreira A., Araújo, J. y Arango F. (2008) . "Traceability Models to Control an Aspectual Model-Driven Development". Proceedings of the 20th International Conference of Engineering & Knowledge Engineering ISBN: 1-891706-22-5 (SEKE 2008).
12. Tabares, M.S. Alferez, G.H., Alferez E.M. (2008). "El Desarrollo de Software Orientado a Aspectos: Un Caso Práctico par a un Sistema de Ayuda en Línea. Revista Avances en Sistemas e Informática Vol. 5, # 2 Junio 2008. ISSN:1657-7663, págs: 61-68.
13. Tabares, M.S., Pineda J.D., Barrera, A.F. (2008). "Un Patrón de Interacción entre Diagramas de Actividades UML y Sistemas WorkFlow". Revista EIA No. 10, Diciembre 2008. ISSN: 1794-1237.
14. Tabares, M.S., Arango, F. (2009). "Achieving Consistency and Completeness of Business Process Models throughout the Lifecycle". Proceedings of the IDEAS International Conference. Colombia Abril 2009. ISBN: 978-978-44-5028-9.

Capítulo 2
Trazabilidad para la Consistencia y la Completitud: una revisión del estado de la técnica

Este capítulo presenta una revisión de la literatura de investigación que trata la trazabilidad en el contexto de la ingeniería de software como la base para validar y verificar la consistencia y la completitud de los requisitos y modelos de un sistema. Para lograr esto se desarrollan dos objetivos: (1) definir la trazabilidad, la consistencia y la completitud desde sus conceptos básicos, técnicas y características generales en su tratamiento; (2) evaluar la forma como diferentes enfoques proponen la práctica de la trazabilidad como medio para lograr atributos de calidad tales como la consistencia y la completitud.

Los enfoques se agrupan en contextos de la ingeniería de software que facilitan la evaluación ya sea porque su único objetivo es el tratamiento de la trazabilidad como son los modelos orientados a la trazabilidad, o porque son nuevos paradigmas de desarrollo que proveen elementos o mecanismos que favorecen la trazabilidad, como son los modelos de desarrollo orientados a aspectos y los modelos orientados al desarrollo dirigido por modelos. Además se hace una evaluación de los modelos adoptados por la industria del software para tener una visión real del tratamiento de la trazabilidad. De cada contexto se analizan los enfoques más representativos en el tratamiento de la trazabilidad y se determina el aporte que hacen para lograr la consistencia y la completitud de los requisitos a través de los modelos de desarrollo.

Como resultado del análisis se discuten aspectos relevantes a la práctica que proporcionan los diferentes contextos y se proponen varios escenarios de solución que podrían ayudar en la adopción y uso de la trazabilidad como un medio formal para las tareas de verificación.

2.1 Introducción

Cuando se habla de trazabilidad de requisitos en los equipos de desarrollo de desarrollo de software nadie niega la importancia que tiene esta práctica para lograr los objetivos de un proyecto y generar productos de calidad. Sin embargo, muchos de ellos desconocen como optimizar su uso para generar valor agregado durante todo el proyecto de desarrollo del software.

La trazabilidad es un atributo de calidad importante cuya práctica debe soportar transversalmente el proceso de desarrollo. Es parte activa de la gestión del cambio o gestión de la configuración del software, por lo tanto debe soportar el control de versiones, el seguimiento de la evolución de los requisitos en modelos de desarrollo, la propagación del cambio, y el análisis de impacto del cambio.

Generalmente la práctica de la trazabilidad se soportada en herramientas CASE (*Computer-Aided Software Engineering*) para el modelado de requisitos y modelos de software las cuales estandarizan la práctica por medio de matrices de trazado y relaciones de abstracción UML que facilitan la construcción de modelos de trazabilidad. Las matrices de trazado facilita el cruce de los requisitos con elementos de modelo que los implementan, ayudan a resolver conflictos entre los participantes directos o indirectos de la aplicación (*stakeholders*) y logran ser un artefacto que facilita el entendimiento y negociación entre usuario y analistas. Sin embargo, las matrices se usan principalmente en la etapa de requisitos y en la etapa de pruebas, quedando rápidamente desactualizadas durante las etapas intermedias del proceso de desarrollo. Los modelos de trazabilidad es una buena forma de mantener la trazabilidad pero muchas veces no se usan correctamente debido a las diferentes interpretaciones dadas a los vínculos de trazado, o la forma como se combinan con los modelos de desarrollo.

Es posible ver en las empresas de desarrollo que la práctica de la trazabilidad no se usa en todos los proyectos, ni por todos los equipos de desarrollo durante el proceso de desarrollo, aún así sea una actividad de calidad establecida en el proceso. También se evidencia el desconocimiento de otros enfoques de trazabilidad proporcionados por grupos de investigación que podrían dar valor agregado a la calidad del proceso y los productos de software. No es usual que esta práctica se use formalmente para verificar la consistencia y la completitud de los modelos, o medir el impacto del cambio en elementos y modelos hechos durante todo el proceso de desarrollo.

Para dar mayor claridad a los conceptos que se tratan durante esta revisión y hacer un análisis claro y objetivo, este capítulo está organizado de la siguiente forma. La Sección 2.2 presenta los conceptos generales de trazabilidad, consistencia y completitud. La Sección 2.3 presenta otros estudios relacionados con la trazabilidad y define el problema. La Sección 2.4 describe el procedimiento metodológico para aborda la evaluación de la literatura de investigación. La Sección 2.5 presenta el análisis de los diferentes enfoques seleccionados para lograr el objetivo del capítulo. La Sección 2.6 presenta una discusión alrededor del soporte que los nuevos enfoques proveen para el tratamiento de la trazabilidad. La Sección 2.7 presenta los escenarios de solución que guiarán el desarrollo de esta disertación. Finalmente, la Sección 2.8 presenta las conclusiones del capítulo.

2.2 Conceptos Generales de Trazabilidad, Consistencia y Completitud

Esta sección presenta la trazabilidad desde su definición, a través de algunos aspectos generales de aplicación y uso, hasta la visión general de los mecanismos de trazado más conocidos por los desarrolladores para realizar esta práctica.

2.2.1 Definición de Trazabilidad

En Ingeniería del Software, la trazabilidad es un atributo de calidad que establece un conjunto de características y elementos para hacer el seguimiento de la vida de los requisitos durante el proceso de desarrollo. Esta práctica implica realizar actividades de validación y verificación durante el proceso de desarrollo para lograr características como la confiabilidad y la exactitud de los productos de software que se producen.

En particular, la trazabilidad es una práctica de la Ingeniería de Requisitos que facilita el control de los requisitos por medio de vínculos de trazado entre diferentes artefactos que los representan a través del ciclo de vida de desarrollo. Algunas de las definiciones más populares son:

- IEEE define la trazabilidad como “el grado en el cual una relación puede ser establecida entre dos o más productos del proceso de desarrollo, especialmente productos que tienen relaciones de predecesor-sucesor o maestro-subordinado entre uno y otro” [IEEE-STD-610].
- El modelo de madurez CMMI[1] (*Capability Maturity Model Integration*) se define que cuando los requisitos están bien administrados, es posible mantener la trazabilidad bidireccional entre los requisitos y sus productos de trabajo. Es decir, la trazabilidad ayuda

[1] www.sei.cmu.edu/cmmi/

a determinar si todos los requisitos fuente han sido completamente diseccionados y que todos los requisitos de niveles inferiores pueden ser trazados a una fuente válida. De igual forma, puede cubrir las relaciones entre otras entidades tales como productos de trabajo intermedios y finales, cambios en la documentación del diseño y planes de prueba. Esta práctica es necesaria para conducir la evaluación del impacto del cambio en las actividades del proyecto y los productos de trabajo [CMMI-DEV 2006].

- La trazabilidad bidireccional se define en términos de trazabilidad horizontal y vertical. Trazabilidad horizontal se refiere a la posibilidad de vincular los productos de trabajo de subsecuentes fases de desarrollo (relación predecesor-sucesor). Trazabilidad vertical es la posibilidad de vincular productos de trabajo dentro de la misma fase de desarrollo (relación maestro-subordinado) [Pfleeger and Atlee 2005].
- Gotel, se refiere a la trazabilidad como "la habilidad para describir y seguir la vida de un requisito en ambas direcciones hacia delante (forward) y hacia atrás (backward) , i.e., desde su origen, a través de su desarrollo y especificación, hasta su construcción y uso, y a través de todos los períodos de refinamiento en curso e iteración en alguna de estas fases". Además, define la trazabilidad de los requisitos desde dos vistas. (1) pre-trazabilidad (Pre-RS[2]), la cual hace referencia a los aspectos de la vida del requisito desde antes de incluirlos en la especificación de requisitos hasta las fuentes (documentos de procesos de negocio y usuarios). (2) post-trazabilidad (Post-RS), la cual hace referencia a los aspectos de la vida de los requisitos desde su inclusión en la especificación de requisitos y a través del diseño hasta el código [Gotel and Finkelstein 1994].
- Clarke, define esta práctica como "la habilidad para determinar realmente como una pieza de un artefacto de software (requisito, diseño, código) afecta a otros. La trazabilidad hace posible buscar un cambio en un requisito y encontrar sus partes en los detalles del diseño y el código que son afectados por el cambio. Es esencial para mantener los documentos de requisitos y diseño actualizados al día con respecto a la evolución del código" [Clarke 2002].

Diferentes enfoques de trazabilidad se han propuesto para garantizar la correlación entre los requisitos y los artefactos que los representan en el proceso de refinamiento del ciclo de vida. En general, cada enfoque propone una semántica de trazado diferente que corresponde a la definición de los elementos de trazado o rastreo (trace elements) y vínculos de trazado (tracing links) en el marco de un modelo de trazabilidad conceptual. El significado que sea dado a cada uno de esos

[2] *Requirement Specification*

elementos permitirá hacer el seguimiento a los requisitos desde su identificación hasta su implementación.

Los enfoques de trazabilidad coinciden en soportar el trazado desde tres elementos básicos: los stakeholders, las fuentes (documentos y modelos), y los objetos o artefactos a ser trazados (elementos de los modelos). Además, es común ver que los enfoques orientan la práctica de la trazabilidad en tres direcciones: hacia los participantes, hacia los artefactos, o ambos [Gotel and Finkelstein 1997], [Lindvall 1994], [Ramesh and Jarve 2001], [Egyed 2003], [Cleland-Huang et al. 2003], [Aizenbud-Reshef et al. 2005].

Durante el proceso de desarrollo, los participantes de un proyecto de software determinan la validez de los requisitos. Ellos son responsables por los artefactos construidos durante el proceso, por lo tanto deben toma decisiones que garanticen confiabilidad y exactitud de los productos software. No obstante, controlar la práctica de la trazabilidad durante el proceso muchas veces es difícil porque los participantes tienen diferentes intereses en el sistema. Ramesh expresa que "muchos participantes diferentes — patrocinadores del proyecto, directores del proyecto, analistas, diseñadores, mantenedores y usuarios finales — están involucrados en el ciclo de vida de desarrollo del sistema. La trazabilidad necesita de estos participantes debido a las diferencias en sus metas y prioridades…". Los participantes representan agentes involucrados en el sistema de desarrollo, manejan las fuentes y mantienen las actividades del ciclo de vida [Ramesh and Jarve 2001].

Gotel, realizó un estudio acerca del comportamiento de los participantes con respecto a la práctica de la trazabilidad y encontró varios problemas, entre los cuales están: (i) la falta de acuerdo entre los participantes en cuanto a la cantidad y el tipo de la información que querían trazar con respecto a los requisitos, (ii) el uso de métodos de desarrollo informales, (iii) el fracaso de seguir prácticas estándares, insuficientes recursos (tiempo y personal de apoyo) asignados para realizar la trazabilidad, y (iv) la falta de claridad acerca de papeles desempeñados por participantes en el proceso de trazabilidad. A partir de estos concluye que para lograr la trazabilidad de requisitos es necesario incrementar la conciencia de las necesidades de los participantes en el proceso de desarrollo para identificar los diferentes tipos de información y las relaciones que deberían ser mantenidas [Gotel and Finkelstein 1994, 1997].

Los enfoques que solo incluyen los artefactos en el trazado se orientan a su verificación para asegurar que éstos se implementen correctamente. Esta orientación se debe soportar en marcos de trabajo (*frameworks*) que provean vínculos de trazado y métodos de control sobre los artefactos que varían durante el proceso de desarrollo.

La trazabilidad orienta los grupos de trabajo a tomar decisiones acerca del alcance del desarrollo y el impacto del cambio. Su práctica, en la actualidad, depende de los criterios de calidad establecidos por las empresas de desarrollo o por las facilidades que provean las herramientas CASE para el modelado de los requisitos. Además, nuevos paradigmas de desarrollo proveen enfoques para el tratamiento de los requisitos, su modelado en artefactos de software y su trazabilidad. Algunos de ellos son:

- El Desarrollo Dirigido por Modelos MDD (*Model-Driven Software Development* [Stahl and Volter 2006]) soporta la transformación desde modelos fuente hacia modelos destino en diferentes niveles de abstracción basados en la arquitectura dirigida por modelos MDA[3] (*Model-Driven Architecture*).
- El desarrollo de Software Orientado por Aspectos AOSD[4] (*Aspect-Oriented Software Development*) soporta la separación de asuntos y asuntos transversales en el ciclo de vida de desarrollo (más detalle en las siguientes secciones de este capítulo y en el Capítulo 5).
- El desarrollo por Líneas de Productos de Software SPL[5] (*Software Product Lines*) soportan el desarrollo de productos de software en el ciclo de vida de desarrollo atendiendo especialmente las características y los elementos comunes y variables a diferentes modelos en un dominio específico (la trazabilidad desde este paradigma no es objetivo de esta disertación).

2.2.2 Mecanismos Tradicionales para realizar Trazabilidad

- **Matrices de Trazabilidad**

Es la técnica más usada para realizar trazabilidad en ambientes reales de desarrollo. Las matrices establecen la relación entre requisitos y elementos de modelo de diferentes diagramas y artefactos en diferentes niveles de abstracción. Por ejemplo, la Figura 3 muestra una de las matrices más comunes para validar la trazabilidad de los requisitos: Requisitos vs. Casos de Uso. Otros tipos de matrices de trazado de uso frecuente son: Requisitos vs. Características del sistema/productos observables, Requisitos vs. Fuentes de información, Requisitos vs. Requisitos, Requisitos vs. Subsistemas que los contienen, Requisitos vs. Interfaces internas y externas del sistema [Pressman 2006].

[3] Más detalle se define en las siguientes secciones de este capítulo y en el Capítulo 3 (www.omg.org/mda/).
[4] www.aosd.net
[5] www.sei.cmu.edu/productlines/

Casos de Uso \ Requisitos	CU1	CU2	CU3	CUn
Requisito 1	✓			
Requisito 2		✓		
Requisito 3			✓	
...				...
Requisito n	✓			

Figura 3. Ejemplo de una matriz de trazado: Requisitos vs. Casos de Uso.

Normalmente, las herramientas CASE de modelado soportan la generación automática o semiautomática de las matrices de trazabilidad. No obstante, esta práctica puede ser costosa y tediosa para sistemas complejos o modelos de gran tamaño, dificultando la verificación de la consistencia y la completitud de los requisitos en los modelos de desarrollo.

La matriz CRUD (*Create*, *Retrieve*, *Update*, y *Delete*) es otro tipo de matriz muy usada en la fase de diseño. Esta permite analizar las operaciones que se deben realizar sobre la base de datos a partir de la correlación entre tablas y funciones, u otros elementos del diseño de la base de datos. La construcción de estas matrices trae beneficios más allá de un simple registro de la correlación o dependencia entre los elementos de los modelos. A partir de ellas es posible analizar o inferir información acerca del nivel de especificación de los requisitos, el nivel de participación de los usuarios, el costo asociado a cada fase de desarrollo, la arquitectura requerida, el plan de las pruebas, etc. Si las matrices son actualizadas continuamente, pueden usarse como herramienta de soporte para el personal encargado del mantenimiento y las pruebas del sistema.

Elementos de Trazado UML

El lenguaje de modelado unificado UML[6] (*Unified Modeling Language*) provee la relación de dependencia de forma especializada en la relación de abstracción (*abstraction*) para especificar la trazabilidad de los requisitos y elementos de modelo donde uno de ellos representa una especificación (el proveedor), y el otro representa su implementación (el cliente). Una relación de abstracción "relaciona dos o más elementos o conjunto de elementos que representan el mismo concepto en diferentes niveles de abstracción o desde diferentes puntos de vista" y usa los

[6] www.uml.org/

estereotipos `<<refine>>` y `<<trace>>` [UML-Superstructure 2005]. A su vez la abstracción se especializa en la relación de realización (*Realization*) la cual trata la trazabilidad como la implementación de un elemento en otro.

UML 2.x y algunas herramientas de modelado CASE proveen las anteriores relaciones de abstracción y aún otras tales como: `<<realize>>`, `<<derive>>`, `<<substitute>>`, `<<call>>`, `<<send>>`, and `<<instantiate>>`. En los siguientes párrafos se definen las relaciones de abstracción más usadas durante el modelado: *refine*, *trace* y *realize*.

- **<<refine>>**. UML define la relación de traza Refine para especificar un refinamiento entre elementos de modelo en diferentes niveles semánticos tales como análisis y diseño. La correlación (*mapping*) especifica la relación entre los dos elementos o conjuntos de elementos. Esta puede o no ser computable y puede ser unidireccional o bidireccional. El refinamiento puede ser usado para las transformaciones de modelos y el manejo del cambio en diferentes niveles de abstracción [UML-Superstructure 2005]. También, la relación de traza *Refine* se puede usar entre elementos del mismo modelo. Por ejemplo, se pueden tener dos versiones de una clase en un modelo, uno de los cuales se optimiza para la ejecución [Arlow and Neustad 2005].
- **<<trace>>**. UML especifica la relación de traza entre elementos de modelo o conjunto de elementos de modelo que representan el mismo concepto en diferentes modelos. Este tipo de relación se usa principalmente para rastrear requisitos y cambios a través de modelos. Porque los modelos cambian, puede ocurrir en ambas direcciones (así la dirección de la dependencia se puede ignorar). La correlación especifica la relación entre dos, pero raramente es computable y usualmente es informal [UML-Superstructure 2005]. En este tipo de relación el proveedor y el cliente representan el mismo concepto pero ellos están en diferentes modelos. También, la relación se podría usar para mostrar una relación entre un requisito funcional y el caso de uso que soporta dicho requisito [Arlow and Neustad 2005]. En la Figura 4 se ilustran el uso de la abstracción *trace*.
- **<<realize>>**. Esta relación de traza usa el estereotipo <<realize>> y se podría usar para el refinamiento o correlación de implementación paso a paso durante la optimización, transformación, composiciones, marcos de trabajo, plantillas (*templates*), etc. [UML 2004].

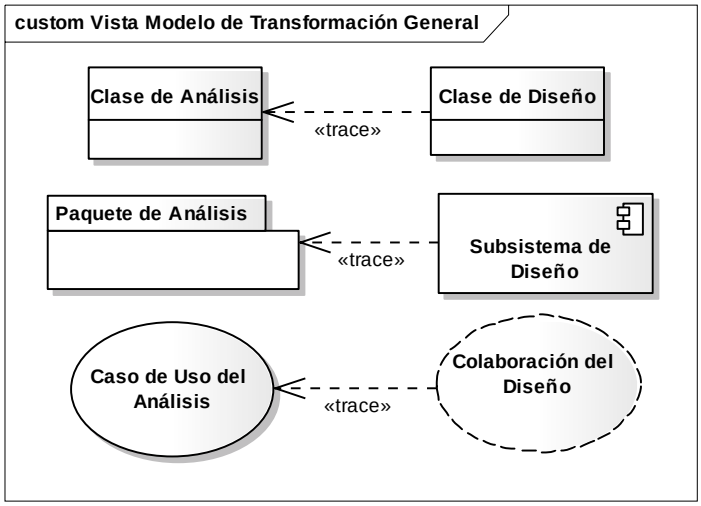

Figura 4. Relaciones de abstracción <<trace>> entre elementos de modelo UML del análisis y el diseño [Arlow and Neustad 2005].

Modelos de Trazabilidad

Los modelos de trazabilidad se construyen con las relaciones de abstracción proporcionadas por UML o definidas por los desarrolladores, con estereotipos que le dan un significado especial al trazado. Los elementos de modelo involucrados en la traza se definen a criterio del grupo de desarrollo así como su utilidad y mantenimiento. Por medio de los elementos de modelo y la semántica de trazado proporcionada por UML es posible crear modelos para realizar la trazabilidad de los requisitos. La Figura 5 ilustra un ejemplo de un modelo de trazabilidad donde el elemento de modelo `UseCase-N` realiza los elementos de modelo `Requirement-A` y `Requirement-B`. Además, este es realizado por los artefactos `Class Diagram` y `Screen-N`. De igual forma, se puede establecer una relación `<<trace>>` entre `UseCase-N` y el elemento de cambio `Change-X`.

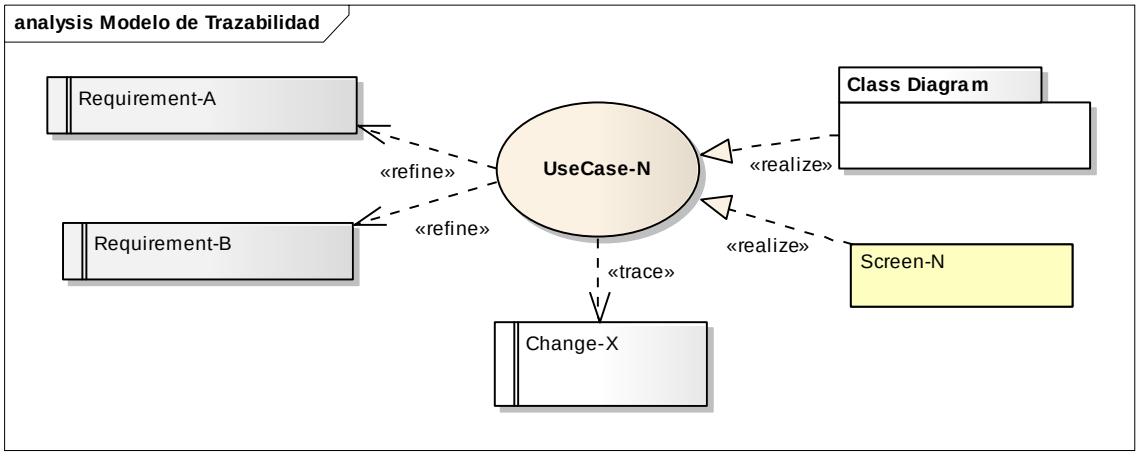

Figura 5. Un ejemplo del uso de las relaciones de trazado proporcionadas por UML en un modelo de trazabilidad.

Los modelos de trazabilidad facilitan la propagación y la estimación del impacto del cambio de acuerdo al modelo de trazabilidad definido. Para los modelos de trazabilidad centrados en casos de uso, comúnmente se aplica el método basado en puntos de casos de uso [Carroll 2005]. Para modelos de trazabilidad que sólo trazan elementos del diseño, la estimación del esfuerzo se hace desde el número de vínculos de trazado, de clases, de componentes y de métodos [Ahn and Chong 2006]. El buen uso de los modelos de trazabilidad dependerá del grado de mantenimiento que estos tengan.

Herramientas de Soporte a la Trazabilidad

Los desarrolladores construyen los modelos de trazabilidad de acuerdo al proceso de desarrollo, las estrategias de calidad planteadas por la compañía de desarrollo, y en gran parte guiada por las características de trazado que provean las herramientas de modelado. Por lo general, las herramientas de soporte automatizado para la trazabilidad, son herramientas que soportan la ingeniería de los requisitos. Algunas de las más reconocidas en la literatura son: Enterprise Architect™-UML CASE Tool[7], Requisite Pro™ Tool[8] [IBM-RRP 2006], DOORS™ (Dynamic Object-Oriented Requirement System)[9] [Azelborn 2000], y otras que están catalogadas en [SE Tools 2004]. Estas herramientas normalmente soportan la trazabilidad de requisitos y sus artefactos por medio de matrices que relacionan elementos de modelo y generan reportes especializados para el control del trazado.

[7] http://www.sparxsystems.com.au/ea.htm
[8] http://www-01.ibm.com/software/awdtools/reqpro/
[9] www.telelogic.com/Products/doors/doors/index.cfm

2.2.3 Consistencia y Completitud de los Requisitos y Modelos

Durante la gestión de la configuración la trazabilidad se convierte en la práctica de control que facilita la verificación y la validación de la exactitud de los requisitos durante el proceso de desarrollo. Por lo tanto, es necesario establecer elementos, restricciones y estrategias para proveer cuatro criterios básicos de calidad: consistencia, completitud, viabilidad y comprobabilidad. Para efectos de cumplir el objetivo de la tesis sólo se definen a continuación la consistencia y la completitud.

- **Consistencia**

La consistencia de los requisitos consiste en mantener su significado a través de los modelos que los representan en diferentes niveles de abstracción. Una teoría es consistente si ésta nunca prueba una contradicción. Formalmente se define que "un cálculo es consistente si y sólo si no es posible demostrar en él una fórmula y su negación" [Deaño 1975]. La consistencia en el desarrollo de software puede tratarse desde dos puntos de vista: consistencia semántica y consistencia entre elementos de modelo.

Consistencia semántica es la hipótesis de reconocimiento del discurso correctamente estructurada en el nivel semántico abstracto [Gurevych et al. 2002]. Por lo general, este tipo de consistencia se trata desde la semántica utilizada para describir los requisitos o para verificar que los elementos de un mismo modelo sean coherentes, por ejemplo, los elementos de un diagrama de estados. Este tipo de consistencia se logra al establecer formas de descripción de los requisitos e identificación gramatical de diferentes elementos que pueden transformarse en elementos de modelo.

La consistencia entre elementos de modelos que representan el mismo requisito o asunto de un sistema es la hipótesis de reconocimiento del discurso correctamente representada en elementos de diferentes modelos. Este tipo de consistencia se trata más comúnmente desde los enfoques de trazabilidad. La consistencia de los requisitos y los modelos se puede verificar, por ejemplo, por medio de fórmulas bien formadas en la lógica formal, matrices de trazado u otros métodos. Pero siempre es importante responder la siguiente pregunta de forma exacta y sin lugar a imprecisión ¿Qué tanto puede mantenerse el significado de los requisitos a través de los modelos que los representan, sin que se generen conflictos o contradicciones entre ellos?

Al hacer una verificación de consistencia es posible clasificar los resultados en tres niveles:

- Consistencia total: ocurre cuando todos los elementos que representan un requisito, interés u objetivo del sistema mantienen las relaciones y el significado entre ellos.

- Semi-consistencia: ocurre cuando parte de los elementos que representan un requisito, interés u objetivo del sistema mantienen las relaciones y el significado entre ellos.
- Inconsistencia: ocurre cuando los elementos que representan un requisito no están semánticamente relacionados dentro del dominio del modelo.

Boehm define que una especificación de requisitos y diseño es consistente en el grado que sus provisiones no generen conflicto entre ellos o con especificaciones de gobierno y objetivos [Boehm 1984]. Las especificaciones requieren consistencia de dos formas: consistencia interna donde los ítems de una especificación no tienen conflictos unos con los otros; consistencia externa donde los ítems de una especificación no tienen conflictos con especificaciones externas o entidades [Boehm 1984].

Engels define consistencia como la propiedad donde submodelos diferentes no se contradicen entre sí. Es decir, existe una realización que satisface todo los requisitos expresados por estos submodelos diferentes [Engels et al. 2002]. Esta incluye consistencia vertical y horizontal. Consistencia vertical se refiere a la consistencia entre modelos de diferentes niveles de abstracción (por ejemplo, un diagrama de clases en el nivel de análisis y un diagrama de clases en el nivel de diseño). Consistencia horizontal, se refiere a la consistencia entre submodelos diferentes en el mismo nivel de abstracción (por ejemplo, un diagrama de clases y un diagrama de estados en nivel de diseño), y consistencia de evolución entre versiones diferentes del mismo submodelo [Engels et al. 2002].

La consistencia vista desde las inconsistencias de los modelos durante el desarrollo es tratada por diferentes investigadores. Nuseibeh et al. definen una inconsistencia como cualquier situación donde un conjunto de descripciones no obedecen algunas relaciones que debería mantenerse entre ellas [Nuseibeh et al. 2000]. Spanoudakis define una inconsistencia como un estado en el cual dos o más elementos que se superponen de diferentes modelos de software hacen afirmaciones sobre aspectos del sistema que ellos describen que no se satisfacen conjuntamente [Spanoudakis and Zisman 2001]. Las técnicas de manejo de inconsistencias ayudan a administrar la evolución de los requisitos orientados a aspectos e incluye un alto manejo del cambio de la información aspectual en vistas, historia del cambio para todas las vistas, componentes y sus aspectos [Grundy 1999].

Olsson y Grundy determinan que las inconsistencias en el desarrollo de software se deben a que un artefacto puede ser representado en diferentes modelos, de forma diferente y en diferentes niveles de abstracción. Además otro tipo de información anexa a los artefactos puede traslaparse, e.g., identificación de usuarios/actores, funciones/acciones, datos de entrada y salida, restricciones no funcionales, y condiciones especiales. Para un buen manejo de los artefactos durante el desarrollo, se debe soportar en la trazabilidad y el manejo de inconsistencias entre vistas que informan a

desarrolladores de relaciones claves y donde los cambios de un modelo han afectado a otro [Olsson and Grundy 2002].

Lange trata la consistencia en los diagramas UML. Si dos diagramas en el sistema contienen información común contradictoria, i.e., ellos describen decisiones de diseño contradictorias, entonces hay inconsistencia entre los dos diagramas [Lange 2003].

Egyed presenta una herramienta para verificar la consistencia de modelos UML. La herramienta, UML/Analyzer[10], está integrada al IBM Rational Rose™ y provee un conjunto de reglas de consistencia para verificar posibles inconsistencias en modelos UML del diseño. Las reglas de consistencia UML tales como sintáctica bien formada, coherencia entre diferentes diagramas y coherencia entre diferentes modelos, describen condiciones que un modelo UML debe satisfacer para que este sea considerado un modelo válido [Egyed 2006].

Goknil trata la consistencia de los requisitos desde el punto de vista a varias restricciones de las relaciones entre ellos. Estas se pueden especificar con base en la formalización de las relaciones para los requisitos del metamodelo. Las inconsistencias indican que las relaciones entre requisitos están violando sus restricciones [Goknil 2008].

Algunas herramientas CASE para el modelado de UML, tales como Together®[11] by TogetherSoft™ [Tog 2007] y ArgoUML[12] [Robins et al. 2006] proporcionan capacidades de trazado y consistencia entre modelos de diseño y código.

Completitud

Formalmente se define que "un cálculo es completo cuando en él se pueden demostrar todas las fórmulas verdaderas construibles con sus símbolos" [Deaño 1975]. La completitud y la consistencia se complementan entre si pero una no asegura la otra. El primer teorema de incompletitud de Gödel dice que cualquier teoría generada efectivamente, capaz de expresar aritmética elemental no puede ser tanto consistente como completa. En la lógica, completitud es la opuesta de la solidez para sistemas formales. Un sistema formal tiene "completitud" cuando todas las tautologías son teoremas mientras que un sistema formal tiene "solidez" cuando todos los teoremas son tautologías [Meltzer 1962].

Desde la ingeniería del software, la completitud es el conocimiento acerca de que cualquier elemento de entrada en los modelos del problema deberá ser incluido en los modelos de la solución.

[10] http://www.alexander-egyed.com/tools/uml_analyzer_tool.html
[11] http://www.borland.com/us/products/together/index.html
[12] http://argouml.tigris.org/

La completitud no necesariamente significa que cada elemento producido se conserve, es decir, si el desarrollador determina que algún elemento de un modelo no tiene sentido, dichos elementos deben ser eliminados de los modelos que los representan sin generar incompletitud en el sistema. Es decir si el sistema está en desarrollo, se debe verificar las relaciones de completitud establecidas entre los modelos y si está en producción es necesario identificar con antelación como impacta el cambio para no generar errores o defectos en el sistema en ejecución.

La completitud puede ser vista desde diferentes puntos de vista. Desde la programación, la completitud computacional es definida y se basa en el teorema de Colomb donde, "para cada función computable parcial `f: O1 → O2` (donde `O` es un conjunto de objetos) hay un programa que compute a `f`" [Colomb and Weber 2003]. Desde el grado de robustez de especificación de los modelos, la completitud se garantiza a partir de una respuesta que se especifique para cada entrada y secuencia de entradas posibles. La verificación de completitud se define como la especificación que es suficiente para distinguir el comportamiento del software deseado de cualquier otro programa indeseado que podría ser diseñado. Especificaciones no determinadas o complejas a menudo esconden peligrosos estados de incompletitud. Desde este punto de vista, es necesario conocer, por cada tipo de modelo, los elementos y sus condiciones de funcionamiento y relación con otros modelos [Heimdahl and Leveson 1996].

Lange define que un modelo UML está completo si cada elemento en el diagrama tiene una correspondencia de otro elemento en otro diagrama, i.e., no hay huérfanos ni hojas sueltas [Lange, 2003].

Verificar la completitud a partir de la transformación de los requisitos garantiza que un conjunto de elementos definidos en el modelo del problema contiene toda la información correspondiente para interpretarlo. Cualquier elemento de entrada necesario para crearlo deberá ser incluido en los modelos de la solución. Por ejemplo, en la Figura 5 se podría decir que el modelo es completo ya que se encuentran todas las entradas registradas para el elemento de modelo `Requirement-A`. Cuando hay cambios es necesario verificar que los elementos de modelo que representan una funcionalidad específica se mantengan en el mismo nivel de abstracción o entre otros modelos de diferente nivel de abstracción. Al verificar la completitud de los requisitos en los modelos podrían formularse preguntas tales como: ¿Hay algún hueco obvio, inexplicado delos artefactos que deberían constituir una funcionalidad específica? ¿Hay artefactos nombrados que no se relacionan con ningún otro, en el mismo o diferente nivel de abstracción?

2.3 Estudios acerca de la Trazabilidad

Para identificar la problemática actual, en esta sección se describen algunos enfoques que han realizado estudios de la trazabilidad ya sea para conocer su uso y adopción en ambientes de desarrollo, o simplemente para conocer el estado de la técnica.

Gotel y Finkelstein presentan un estudio enfocado a conocer el significado de la trazabilidad para diferentes tipos de participantes de un proyecto, el conocimiento y la profundidad que se puede lograr en la práctica a partir del uso de diferentes tipos de trazabilidad (pre y post trazabilidad) [Gotel and Finkelstein 1994].

Ramesh, presenta un estudio donde demuestra que el problema de adopción y uso se centra en cómo se realiza la práctica de la trazabilidad. Así, los artefactos u objetos a ser trazados son diferenciados de acuerdo al tipo de usuario: low-end los cuales usan esquemas simples de trazabilidad y el high-end los cuales capturan la información relacionada con el proceso como la razón fundamental detrás de varios artefactos y la evolución progresiva de estos [Ramesh 1998].

Gills, presenta un estudio cuyo objetivo es determinar la posibilidad de mejorar la práctica de la trazabilidad en empresas de tecnología de información. En el estudio encontró que no hay cultura ni motivación para adoptar la práctica de la trazabilidad de forma constante y controlada, se considera una práctica difícil de implementar en la vida real. Además, muestra como las empresas usan el concepto de "modelos de trazabilidad" adicionando una fase de manejo del cambio al proceso de desarrollo [Gills 2005].

Chitchyan et al. presentan un estudio completo del tratamiento de la trazabilidad desde los enfoques de la ingeniería de requisitos orientada por aspectos. La capacidad de trazabilidad de esos enfoques fue evaluada desde dos criterios: (1) características que soportan la trazabilidad de requisitos y el cambio a sus fuentes de origen, esto ayuda a entender donde un artefacto o cambio sucederá; (2) soporte a la correlación de requisitos a tipos de artefactos de software de diferentes representaciones y etapa del ciclo de vida. Además, se evalúa el soporte a la evolución de los requisitos, los cambios en sus fuentes de origen y el soporte al mapeo [Chitchyan et al. 2005].

Czarnecki y Helsen hacen un estudio de diferentes enfoques (herramientas) de transformación, los cuales se clasifican de acuerdo con diferentes características de alto nivel cuyos dominios son diferentes opciones de diseño para las transformaciones de modelos. En cada dominio se identifican los puntos de mayor variabilidad ya que determinan la evaluación y pueden asociar o agrupar un conjunto de sub-características para identificar el nivel de manejo o aporte de los enfoques. Las características son: especificación, reglas de transformación, control de la aplicación de la regla,

organización de la regla, relación fuente-destino, habilidad de incremento, dirección, y trazado. Esta última evalúa los mecanismos para registrar aspectos diferentes de la ejecución de la transformación tales como creación y mantenimiento de los vínculos de trazado entre elementos de los modelos de la fuente y el destino [Czarnecki and Helsen 2006].

Aizenbud-Reshef et al., analiza diferentes enfoques MDD desde la óptica de los vínculos de trazabilidad, las posibles inconsistencias y los problemas de consistencia en la propagación del cambio desde enfoques MDD [Aizenbud-Reshef et al., 2006].

Galvão y Goknil analizan la trazabilidad en enfoques orientados a los requisitos, al modelado (metamodelos o marcos de trabajo), y modelos de transformación que generan información de trazabilidad. Además, hace una síntesis de los enfoques de acuerdo con representación de la información de trazabilidad, correlación (*mapping*), escalabilidad, análisis del impacto del cambio, y herramientas de soporte [Galvão y Goknil 2007].

En la mayoría de los estudios presentados previamente, el problema de la trazabilidad como práctica de control de los requisitos de software radica en su uso y adopción durante proceso de desarrollo. Ninguno se centra en esta práctica como la base de conocimiento para lograr otros atributos de calidad como la consistencia y la completitud.

En lo que corresponde al uso, el problema se puede centrar en el desconocimiento de los equipos de desarrollo del valor agregado que esta práctica provee para realizar tareas complementarias de gestión del cambio y la verificación de criterios de calidad de consistencia y completitud. No es usual que el grupo de desarrollo formalice esta práctica para dicho propósito durante todo el proceso de desarrollo, en especial para hacer las tareas de verificación o gestionar el cambio de los requisitos en los modelos de desarrollo.

En lo que corresponde a la adopción, el problema se puede centrar en dos aspectos: uno es el esfuerzo, continuidad y mantenimiento que la práctica exige (de hecho, los métodos ágiles la excluyen como práctica de control por el costo en tiempo que esta genera); el otro es el desconocimiento de mecanismos de trazabilidad que proveen diferentes enfoques para la validación y verificación de los requisitos u otros artefactos del sistema que pueden complementar o reemplazar las tradicionales matrices de trazado. Aunque la trazabilidad normalmente es manual o asistida parcialmente por algunas herramientas CASE, los desarrolladores no logren mantener los vínculos de trazado y los modelos de trazabilidad queden desactualizados con respecto a los modelos de desarrollo.

2.4 Proceso Metodológico para la Evaluación de los Enfoques de Trazabilidad

El proceso metodológico usado para la evaluación del estado de la práctica de la trazabilidad se describe a continuación:

1. Identificar diferentes enfoques que definen esta práctica por medio de modelos, técnicas o mecanismos. Algunos de los elementos que son tenidos en cuenta para la evaluación son vínculos de trazado, reglas de chequeo y elementos trazables que permiten localizar, identificar y clasificar las inconsistencias y la incompletitud de los modelos de desarrollo. Para lograr esto se definen cuatro dominios que facilitan la agrupación de los enfoques de trazabilidad:
   - Modelos tradicionales orientados a la trazabilidad, los cuales proveen mecanismos o métodos generales para realizar la práctica de la trazabilidad de los requisitos en etapas tempranas de desarrollo.
   - El desarrollo de software orientado por aspectos, el cual provee artefactos o modelos que contribuyen al tratamiento de la trazabilidad como un interés transversal.
   - El desarrollo de software dirigido por modelos, el cual contribuyen al tratamiento de la trazabilidad desde la transformación de modelos generando valor agregado para la propagación e impacto del cambio.
   - El tratamiento de trazabilidad desde la industria del software.
2. Evaluar los enfoques más relevantes por medio de un espacio de discusión para establecer una postura con respecto a los alcances de las propuestas y la perspectiva de investigación que se plantea para esta disertación.
3. Definir los escenarios de trabajo por medio de los cuales sea posible cubrir las debilidades que presentan los actuales enfoques para el tratamiento de la consistencia y la completitud desde la práctica de la trazabilidad.

2.5 Evaluación del Estado de la Práctica de la Trazabilidad

Esta sección hace la revisión del estado de la práctica de la trazabilidad como medio para verificar consistencia y completitud de los requisitos durante el ciclo de vida de desarrollo. En cada uno de los puntos de vista enunciados anteriormente se han seleccionado los enfoques de

mayor referencia en esta área del conocimiento. La evaluación identifica el objetivo de cada enfoque, resalta sus principales características y selecciona los elementos, mecanismos o características que facilitan la verificación de la consistencia y la completitud de los requisitos en los modelos de software.

2.5.1 Enfoques Orientados a la Trazabilidad

Los enfoques aquí tratados son enfoques que proponen modelos o mecanismos de trazabilidad independientes a los modelos de desarrollo. Además de sus características generales, se resalta su aporte para lograr consistencia y completitud. Para realizar la evaluación se seleccionaron seis de los enfoques más completos y representativos en el marco de la trazabilidad de requisitos: Lindvall&Sandahl [Lindval and Sandahl 1996], Gotel&Finkelstein [Gotel and Finkelstein 1997], Ramesh&Jarke [Ramesh and Jarke 2001], Egyed [Egyed 2002, 2003, 2004], Cleland-Huang et al. [Cleland-Huang et al. 2003], y Aizenbud-Reshef [Aizenbud-Reshef et al., 2005]. A continuación se describe cada uno de ellos.

Implicaciones prácticas de la trazabilidad. Lindvall y Sandahl proponen vínculos de trazado, no nombrados, entre elementos de modelo del mismo tipo, por ejemplo de `objeto` a `objeto` o de `caso de uso` a `caso de uso`, para especificar el refinamiento de un mismo elemento en diferentes niveles de abstracción (i.e., el dominio, el análisis, el diseño y el código). El trazado se hace por referencia, nombre, o por el conocimiento que el desarrollador tenga del dominio y del sistema para diferenciar otros ítems que estarían relacionados. Los vínculos de refinamiento facilitan la verificación de la completitud, conjuntamente con el nombre de los elementos. El enfoque es limitado ya que no provee un método o reglas para la verificación, no obstante deja la posibilidad de evaluar en el modelo de desarrollo la existencia de algún ítem o conjunto de ítems y de asociaciones que se contradigan entre sí o que de una u otra forma hagan el modelo inconsistente. La inspección es informal y no tiene forma de validar si todos los ítems y asociaciones son consistentes a través de los diferentes artefactos en los cuales se representan.

Aunque la verificación de la consistencia y la completitud no se logran de forma explícita en la propuesta, el enfoque propone un análisis de robustez entre el análisis de requisitos y el diseño. El analista cambia el modelo de análisis de modo que esté listo para ser resistente a los cambios que son más esperados en un futuro previsible. Los objetos son divididos en tres categorías diferentes: entidad, interfaz, y de control, con el objetivo de identificar los objetos según su frecuencia de cambio esperada. Por ejemplo, se supone que objetos de interfaz se cambian más con frecuencia que el resto.

En el enfoque se considera la necesidad y posibilidad de evaluar si el material (documentos) de especificación de requisitos utilizados en los diferentes niveles de abstracción está completo. Es una inspección intuitiva que depende de la importancia que cada stakeholder le dé a los documentos producidos en el ciclo de vida [Lindval and Sandahl 1996].

Trazabilidad de requisitos extendida. Gotel y Finkelstein proponen la trazabilidad de los artefactos conjuntamente con las responsabilidades de los participantes, es decir la consistencia y completitud están asociadas a esto. La consistencia se evalúa informalmente a partir de los compromisos de los participantes (rol social) que se representan en una matriz donde se relaciona cada artefacto con el rol social del participante, los colaboradores y los artefactos que lo adoptan y lo refieren. Así, es posible manejar las posibles contradicciones entre ellos y los elementos de los cuales son responsables. Algunos vínculos de trazado podrían ayudar en las tareas de verificación: *a* `califica a` *b*, *a* `es la razón para` *b*, *a* `define` *b*, *a* asiste con *b*, *a* `es la base para` *b*, *a* `es el resultado de` *b*, *a* `elabora a` *b*, etc., De igual forma los vínculos de trazado entre los artefactos: *a* `adiciona a` *b*, *a* `es marco para` *b*, siendo *a* y *b* dos artefactos del sistema.

En el enfoque se asumen la completitud como la clase de información que puede ser deducida sobre los compromisos de aquellos agentes que han contribuido en el proyecto. Algunos vínculos de trazado podrían ayudar en las tareas de verificación: *a* `es comparada con` *b*, *a* `refina` *b*, *a* `responde a` *b*, *a* `replica a` *b*, *a* `extiende a` *b*. De igual forma los vínculos de trazado entre los artefactos: *a* `justifica` *b*, *a* `asociado con` *b*, *a* `modifica a` *b*, *a* `es causa de` *b* [Gotel and Finkelstein 1997].

Hacia modelos de referencia para la trazabilidad de requisitos. Ramesh y Jarke identifican tres elementos básicos para realizar la práctica de la trazabilidad: las fuentes, los participantes y los objetos a ser trazados. Los participantes representan agentes involucrados en el sistema de desarrollo, manejan las fuentes y mantienen las actividades del ciclo de vida. Ellos tienen diferentes roles o capacidades sobre los objetos y los vínculos de trazabilidad. El modelo de trazabilidad provee dependencias y vínculos de trazado entre los objetos y un meta administrador que garantiza una base de datos basadas en el conocimiento para búsquedas avanzadas de información con respecto al trazado.

La adopción de la práctica de la trazabilidad está influenciada por los siguientes contextos: tecnológico, estrategias corporativas y políticas, y el personal del sistema de desarrollo. La consistencia formal se soporta en la herramienta *ConceptBase* (de uso académico únicamente), la cual proporciona a los "*high-users*" cuatro submodelos en los cuales declaran los elementos de

modelo y las relaciones entre ellos. Los submodelos son: administración de requisitos, racional, diseño/asignación, y verificación de conformidad.

La verificación de la completitud se hace por medio de los `Procedimientos de Verificación de Conformidad` (`CVP`) los cuales son realizados en los componentes del sistema verificando que el componente satisface los requisitos o ayudan a la generación de propuestas de cambio para los requisitos, el diseño, o la implementación. Así, los `CVP` son usados para verificar la completitud y la exactitud del sistema e identificar que los cambios sean necesarios para lograr los objetivos. Si un cambio ocurre en los requisitos, entonces los vínculos de trazabilidad podrían identificar el `CVP` que debe ser modificado o reconstruido [Ramesh and Jarke 2001].

Un enfoque dirigido por escenarios para el análisis de dependencias de traza. Egyed propone un modelo para producir información de trazabilidad que se genera entre los sistemas de software y sus modelos. Básicamente, el enfoque detecta dependencias de traza entre cualquier elemento de modelo que tiene relación con el código. Por ejemplo: clases y métodos en diagramas clases, procesos en diagramas de flujos, acciones y actividades en diagramas de estados, o casos de uso relacionados al código. Este modelo está basado en la observación de escenarios de prueba que son ejecutados durante la ejecución de un sistema de software. A partir de dicha observación se establecen vínculos de trazado entre los elementos de modelos (e.g., clase y flujos de datos) y su correspondiente código fuente. El enfoque básico está orientado a manejar tipos de trazado entre: escenarios y el sistema, elementos de modelo y el sistema, escenarios y elementos de modelo, elementos de modelo y elementos similares, entre los mismos elementos de modelo, entre conjuntos de elementos del modelo, y posibles inconsistencias e incompletitudes. El análisis de trazabilidad es posible ya que los escenarios pueden superponerse en las líneas del código que ellos ejecutan (sus huellas). La traza de escenarios se captura en la forma de una estructura de grafo que tiene tantos nodos como sea necesario representar explícitamente todos los traslapos posibles entre todos los escenarios.

El modelo provee tres dependencias de traza: `hipotetizada`, `generada` y `observada`. La primera actividad requiere analizar y definir las posibles trazas hipotéticas que pueden ser levantadas desde la documentación o los modelos (trazas definidas entre los elementos de modelo que van a estar relacionados con los escenarios). Las trazas generadas a través de una iteración pueden ser usadas como trazas hipotéticas en iteración sucesiva, además usa conocimiento corriente de las trazas y lo permuta con trazas observables. La verificación de la consistencia y la completitud se hace a partir de las trazas hipotéticas excesivas. Las inconsistencias implican que las restricciones conflictivas que se dan como entrada y no como solución podrían satisfacer todas las

restricciones dadas. Las inconsistencias detectadas son: si un elemento de modelo es incluido e excluido en el mismo nodo; si un elemento de modelo no traza a ningún código; si un nodo hoja contiene más de un elemento incluido del mismo grupo; si los elementos de modelo de un padre no son iguales a la suma de todos los elementos de modelo de sus hijos (incluidos + subconjunto); si un nodo hoja (i.e., un elemento de código) no relaciona a algún elemento de modelo de un grupo determinado pero tiene elementos desconocidos de ese grupo.

A partir de las trazas hipotéticas excesivas, Egyed propone la siguiente verificación de completitud: si un elemento del modelo m es exactamente alguna huella de trazado `f` (en el código), entonces se reconoce que `m` traza a dicha huella de trazado y no a otra, además que la huella pertenece a `m` y no a otro modelo. Ésta se considera una entrada precisa y completa. El estado incompleto implica que los datos de entrada fueron insuficientes y la naturaleza exacta de las dependencias de traza no podría ser deducida en todos los casos. Los datos de entrada incompletos conducen a resultados ambiguos. El estado incompleto puede ser medido en como bien los nodos de hoja en el gráfico están relacionados con elementos modelos individuales Egyed [Egyed 2002, 2003, 2004].

Manejo del cambio a partir de eventos identificables. Cleland-Huang et al. define artefactos trazables como aquellos elementos que están débilmente acoplados pero son vinculados a través de un evento de servicio el cual crea un ambiente donde un cambio es manejado más eficientemente y los artefactos y sus vínculos son mantenidos en un estado de restauración. Un artefacto es crítico si este reside como un artefacto intermedio entre múltiples caminos de trazabilidad de requisitos, porque el fracaso de mantenerlo causaría el fracaso de mantener otros artefactos en niveles inferiores en el mismo camino de trazabilidad. Un artefacto está en estado consistente cuando su estado y el estado de sus vínculos exactamente representan el estado actual de la configuración del sistema [Cleland-Huang et al. 2003].

Semántica operacional para trazabilidad. Aizenbud-Reshef et al. definen una semántica de trazabilidad centrada en artefactos UML para facilitar tareas tales como: analizar el impacto y la cobertura de búsqueda, hacer seguimiento a los vínculos de trazado a través del ciclo de vida de desarrollo y mantener el sistema y su documentación a la fecha en estado consistente. Usa las reglas ECA (*Event, Condition, Action*) como forma de proveer consistencia en los modelos UML. Las acciones preventivas son similares al proceso de consistencia vía monitoreo, donde un ambiente de desarrollo (implícita o explícitamente) monitorea el estado de un modelo para asegurar que las acciones hechas por los usuarios (adicionar nuevos elementos o cambiar relaciones) no pueden introducir inconsistencias [Aizenbud-Reshef et al., 2005].

2.5.2 Enfoques Orientados al Desarrollo por Aspectos

La separación de asuntos (*separation of concerns*) es una técnica que permite dividir un problema y caracterizar funcionalidades particulares relacionadas con objetivos del sistema [Dijkstra 1976], [Parnas 1972]. En la separación es posible encontrar que diferentes módulos con comportamiento autónomo necesitan de la colaboración de otros para realizar una función completa en el sistema o simplemente cumplir una serie de restricciones de arquitectura.

El Desarrollo de Software Orientado por Aspectos AOSD (*Aspect-Oriented Software Development*) trata la separación de asuntos y provee un conjunto de enfoques que le permiten al desarrollador establecer criterios de descomposición y composición de asuntos (*concerns*) en diferentes fases del ciclo de vida. En especial, los enfoques orientados a aspectos facilitan la identificación y manejo de artefactos que posiblemente quedarán dispersos (scattered) o enredados (tangled) en otros artefactos o módulos del sistema [Kiczales et al. 2001], [Filman et al. 2004]. Esos artefactos son llamados asuntos transversales (*crosscutting concerns*) ya que actúa como un colaborador que apoya, sin invadir, en las operaciones o actividades básicas de otros asuntos. Por esta razón ellos deben ser compuestos o "tejido" a la funcionalidad de los asuntos base del sistema [Sutton & Rouvellou 2004].

Los aspectos son tratados desde los requisitos hasta la implementación [Clarke and Baniassad 2005], [Jacobson and Ng 2005]. Aspectos tempranos (Early Aspects) identifica, clasifica y modulariza los asuntos y crosscutting asuntos desde el contexto del problema hasta la arquitectura facilitando la trazabilidad de los requisitos y la solución de conflictos principalmente ([Tekinerdogan et al. (Eds.) 2004], [Baniassad and Clarke 2004], [Moreira et al. 2005], [Chitchyan et al. 2005], [Kande 2003], [Tekinerdogan et al. 2004], [Pinto and Fuentes 2007], etc.). Al nivel del diseño y implementación son tratados como estructuras aspectuales que cruzan a otras clases base del sistema [Suzuki and Yamamoto 1999], [Sutton and Rouvellou 2004], [Kiczales et al. 2001].

La trazabilidad se puede ver como un asunto transversal a muchos otros asuntos del sistema ya que tiene las características necesarias para controlar la evolución de los requisitos durante el proceso de desarrollo. Esta práctica se trata la mayoría de veces desde los Aspectos tempranos o la ingeniería de requisitos orientada por aspectos AORE (Aspect-Oriented Requirement Engineering). Algunos de ellos son Theme [Clarke and Baniassad 2005] y AOSD/UC [Jacobson and Ng 2005] definen el tratamiento de los asuntos a través del ciclo de vida.

Aunque la capacidad de trazabilidad de los enfoques orientados a aspectos fue revisada en [Chitchyan et al. 2005], para el objetivo de esta sección se han seleccionado algunos de los enfoques más representativos para analizar la capacidad de verificar la consistencia y la completitud a partir de la trazabilidad de asuntos y asuntos transversales. Los enfoques que se analizan son: Ingeniería de Requisitos Orientada a Asuntos (CORE) [Moreira et al. 2005]; Theme/UML [Clarke and Baniassad 2005]; Análisis de los asuntos transversales a través de fases de desarrollo de software basadas en la trazabilidad [Berg et al. 2006a], Metamodelo para la Trazabilidad de Asuntos [Tekinerdogan et al. 2007]. A continuación se describe cada uno de ellos.

Ingeniería de Requisitos orientada a Asuntos CORE (*Concern Oriented Requirements Engineering*). CORE es un modelo para el tratamiento uniforme de asuntos que enfatiza el tratamiento de asuntos transversales (modelo evolución de AORE con la herramienta *Arcade* [Rashid et al 2003]. Un asunto se definido como un conjunto coherente de requisitos y los asuntos transversales son identificables en matrices donde es posible establecer cuándo algunos de sus requisitos influencian de manera positiva (colabora) o afectan de manera negativa (restringe) otros asuntos. Este enfoque está orientado principalmente a realizar las siguientes tareas: (a) identificar, separar y componer asuntos transversales; (b) soportar el establecimiento de compensaciones (*trade-offs*) tempranas entre asuntos transversales y requisitos que se sobreponen; (c) negociar y tomar decisiones entre grupos de participantes [Moreira et al. 2005]. Las tablas de referencias cruzadas cuya función básica es identificar los asuntos transversales facilitan proveen la información básica para realizar la trazabilidad de asuntos. El Framework PROBE verifica la consistencia de las reglas de composición que se generan a partir del cruce transversal definido en las tablas. Aún así, la consistencia entre modelos no se define [Katz and Rashid 2004].

CORE usa varias tablas de referencia cruzada para identificar impactos y contribuciones de los asuntos. Cuando un cambio ocurre, en cada una de esas tablas la línea/columna cambiada tendrá que ser examinada sin afectar el resto de las tablas. Estas tablas también proporcionan las referencias necesarias para los requisitos cuya composición debería ser examinada.

Theme/UML. Theme/UML propone la separación de asuntos desde el concepto de "tema" (theme). Cada tema declara un elemento de diseño que agrupa una colección de estructuras (clases) y comportamiento (diagramas de secuencia) que representan una característica del sistema. Los temas son un concepto más general que los aspectos y pueden ser relacionados entre sí. La relación de composición especifica cómo los modelos deben ser formados identificando conceptos que se superponen en modelos diferentes y especificando cómo los modelos deberían ser integrados. La consistencia se puede verificar por medio de los vínculos establecidos entre los temas, de tal forma

que se conserve la consistencia si un cambio es realizado en alguno de ellos [Clarke and Baniassad 2005], [Jackson & Sanchez 2006].

Análisis de los asuntos transversales a través de fases de desarrollo de software basadas en la trazabilidad. Berg et al. definen los asuntos transversales basados en un patrón de trazabilidad y las relaciones de dependencia (correlación) entre elementos de la fuente y el destino. Estos casos de correlación son: *injection* (relación 1:1), *scattering* (relación 1:n), *tangling* (n:1), y *crosscutting* (relación n:n). La trazabilidad se provee por medio de las relaciones intra-niveles y entre-niveles (aunque el intra- no se desarrolla en esta propuesta, este denota acoplamiento entre elementos de un nivel de abstracción específico). El análisis de impacto de dependencias transversales se realiza por medio de reglas de transformación al nivel de metamodelo y modelo. Presentan un marco de trabajo para analizar el impacto de cambio en caso de que se presenten relaciones transversales, "crosscuting", en las transformaciones de modelos. El "crosscutting" ocurre cuando se traza una correlación de un elemento de la fuente a dos o más elementos destino y al menos uno de estos elementos destino tiene una correlación con uno de esos elementos de la fuente [Berg et al. 2006a]. Aunque la forma como define la trazabilidad garantiza la consistencia y la completitud, el enfoque no lo define de manera explícita para una posterior verificación de los modelos.

Metamodelo para la trazabilidad de Asuntos. Este enfoque, propuesto por Tekinerdogan et al., propone un metamodelo de trazabilidad desde el modelado de arquitectura de software. Específicamente, logra la trazabilidad de intereses dentro de, y a través de, vistas arquitectónicas, las cuales representan un conjunto de elementos del sistema y relaciones asociadas con ellos. Además, estas pueden tener diferentes tipos de elementos, relaciones y restricciones. Este trabajo se orienta principalmente en el trazado asuntos transversales (aspectos al nivel del diseño) en las vistas arquitectónicas. Esto debido a que los cambios en los intereses como tal pueden tener consecuencias para otros elementos arquitectónicos, que estén directamente o indirectamente relacionados con ellos. Aunque no se hace de forma explícita se podría decir que la consistencia entre vistas arquitectónicas se conserva con la relación de trazado [Tekinerdogan et al. 2007].

2.5.3 Enfoques Orientados a la Transformación de Modelos

El desarrollo dirigido por modelos establece criterios de transformación de modelos separando los artefactos software en modelos identificados en diferentes niveles de abstracción. Este se soporta en la arquitectura dirigida por modelos MDA [Kleppe et al. 2003], y está orientado a proveer información para la trazabilidad de los requisitos basada en las reglas de transformación. En el desarrollo de software dirigido por modelos, la trazabilidad soporta el proceso de transformación al

registrar la ejecución de las transformaciones y proveer información para acciones de valor agregado tales como la propagación y estimación del cambio, la recuperación de la historia de los modelos (backward), entre otras. La trazabilidad tratada desde la transformación de modelos automática o semi-automática en una técnica inherente a cualquier estilo de modelado haciendo más productivo el proceso de desarrollo, facilitando la verificación de la consistencia y la completitud de los modelos para obtener productos de software más confiables y exactos.

La trazabilidad desde la transformación de modelos debe proporcionar como mínimo:

- El tipo de información registrada, por ejemplo los vínculos entre elementos de la fuente y el destino, las reglas que hay creadas entre ellos, y el momento de la creación.
- El nivel de abstracción de la información registrada. Por ejemplo, vínculos sólo para transformaciones de alto nivel.
- El alcance de la información registrada. Por ejemplo, trazar algunas reglas o parte de los elementos de la fuente.
- El lugar donde el vínculo es almacenado. Por ejemplo, en la fuente o el destino o por separado.

Algunos leguajes se han creado para automatizar el proceso de transformación. Por ejemplo, QVT [MOF-QVT 2005], y ATL (Atlas Transformation Language) [Jouault and Kurtev 2005] los cuales proveen mecanismos para la trazabilidad de las transformaciones. Tefkat es un lenguaje declarativo que provee soporte para el trazado donde los desarrolladores deben de crear manualmente los vínculos de trazado para las reglas de transformación [Lawley and Steel 2005]. Otros enfoques como AGG, VIATRA, y GReAT (Graph Rewriting and Transformation language) [Agrawal et al. 2003], no dan soporte a la trazabilidad.

La trazabilidad en el desarrollo de software dirigido por modelos soporta el proceso de transformación al registrar la ejecución de las transformaciones y proveer información para acciones de valor agregado tales como la propagación y estimación del cambio, la recuperación de la historia modelos (*backward*), entre otras. A continuación se evalúan algunos de los enfoques de transformación de modelos más representativos para el objetivo de esta disertación. Algunos de ellos son enfoques no orientados por aspectos y otros enfoques de transformación orientados por aspectos cuyas propuestas aportan directamente a la trazabilidad.

Los enfoques no orientados a aspectos que son: Trazabilidad e interoperabilidad en diferentes niveles de abstracción en la transformación de modelos [Bondé et al. 2005]; Trazabilidad entre modelos de arquitectura de software [Feng et al. 2006]; Trazabilidad de Requisitos y Conformidad

de Transformación en el desarrollo dirigido [Almeida et al. 2006]; Hacia una solución genérica para la trazabilidad en MDD [Walderhaug et al. 2006]; Trazabilidad como entrada para la transformación de modelos [Vanhooff et al. 2007]; Construyendo clasificaciones de trazabilidad en la ingeniería orientada a modelos [Paige et al. 2008]; Aplicación de la transformación de modelos en el marco genérico para trazabilidad [Derezinska and Zawlocki 2008].

Los enfoques orientados por aspectos evaluados son: Análisis de transversalidad en la transformación de modelos [Berg et al. 2006b]; Gestión del cambio basada en trazabilidad en correlaciones operacionales [Kurtev et al. 2007]; Un motor de trazabilidad dedicado a la transformación de modelos para la ingeniería de software [Amar et al. 2008].

Trazabilidad e interoperabilidad en diferentes niveles de abstracción en la transformación de modelos. Bondé et al. presentan un enfoque orientado a solucionar el problema de interoperabilidad en la transformación de modelos usando un modelo de trazabilidad. La trazabilidad en el modelado de software puede ser definida como la capacidad de trazar los elementos de los modelos desde el diseño hacia la implementación. El enfoque provee un metamodelo de trazabilidad que soporta la interoperatividad entre modelos cuando se hacen las transformaciones. El enfoque declara el siguiente principio de consistencia: un modelo de trazabilidad debería ser capaz de identificar todos los elementos en un modelo creado por transformaciones que están relacionadas con un elemento dado en el modelo de la fuente de modo que cuando este se modifique, se pueda propagar el cambio automáticamente o manualmente a todos los modelos relacionados [Bondé et al. 2005].

Trazabilidad entre modelos de arquitectura de software. Feng et al. tratan la trazabilidad entre modelos de arquitectura de software en diferentes fases del ciclo de vida. Esto es, desde modelos de diseño conceptuales, componentes, de implementación y ejecución de la arquitectura. Los modelos de trazabilidad soportan la traza que registran los elementos creados o modificados en los diferentes modelos de arquitectura. Cuando un modelo conceptual se crea, se inicializa un modelo de trazabilidad con toda la información de los artefactos creados; luego, al crearse el modelo para el componente de composición, cada cambio es registrado en el modelo conceptual y otros cambios intermedios son sobrescritos; una vez termine este proceso, los dos modelos son comparados y se eliminan inconsistencias entre ellos. Este proceso se repite hasta lograr el modelo de la arquitectura de ejecución.

Una vez se crea el modelo de la Arquitectura para la composición de componente (destino), el modelo de trazabilidad se verifica contra el modelo de Arquitectura conceptual (fuente) y el modelo de Arquitectura para la composición de componentes, resolviendo de esta forma todas las

inconsistencias. De igual forma se hace con los otros modelos de forma sucesiva y dependiendo el modelo creado o afectado, lo cual aplica a la búsqueda de la completitud [Feng et al. 2006].

Trazabilidad de Requisitos y Conformidad de Transformación en el desarrollo dirigido. Almeida et al., provee un marco de trabajo metodológico que permite a los diseñadores relacionar requisitos a los diferentes productos del proceso de diseño dirigido por modelos. Este marco de trabajo es una base para el trazado de requisitos y la evaluación de la calidad de las especificaciones del modelo de transformación para metamodelos, modelos y realizaciones. La trazabilidad se realiza por medio de una matriz donde se cruzan los requisitos y los modelos de diferentes niveles de abstracción.

Este enfoque asume de primera mano que una especificación de requisito es verificable, es decir, considerando una realización es posible conducir actividades de evaluación para determinar si las exigencias pueden ser consideradas satisfechas. Las actividades de evaluación tales como pruebas de aceptación por usuarios finales, comprobación de modelos, o pruebas de exactitud formales, se usa para indicar el acto de comprobación si un requisito está satisfecho por los modelos (completitud). Así, un modelo que satisface todos los requisitos puede ser considerado una realización. La completitud se verifica al nivel de los modelos de diferentes niveles de abstracción desde el concepto de conformidad entre modelos de la siguiente forma: un modelo MT conforma a otro modelo MS si y sólo si, el conjunto de realizaciones aceptables para MT está contenido en el conjunto de realizaciones aceptables para MS (siendo MT y MS modelos en diferentes niveles de abstracción). Si un modelo en un nivel inferior de la abstracción (M_{i+1}) no se conforma a un modelo en el nivel anterior (M_i), un diseñador está obligado a considerar tanto M_{i+1} como Mi en un paso de diseño siguiente (refinamiento) [Almeida et al. 2006].

Hacia una solución genérica para la trazabilidad en MDD. Walderhaug et al. presentan un enfoque centrado en los artefactos y los tipos de relaciones usadas para la trazabilidad de tal forma que la información del trazado se pueda compartir entre herramientas, organizaciones — proyectos, y productos del ciclo de vida. El modelo propuesto, está orientado a cumplir los siguientes objetivos: definir un modelo de trazado flexible que permita la personalización de los artefactos y relaciones; soportar todo el ciclo de vida de los artefactos; soportar el trazado de los artefactos a través de diferentes herramientas. Para lograr esto proveen un conjunto de servicios tales como: administración del modelo de trazado, creación de la traza, uso y monitoreo del trazado para soportar cualquier tipo de artefacto y relación en ambientes heterogéneos de MDD. El enfoque adopta el tratamiento de la consistencia dada por [Aizenbud-Reshef et al., 2005] pero no especifica nada al respecto [Walderhaug et al. 2006].

Trazabilidad como entrada para la transformación de modelos. Vanhooff et al. presentan un enfoque donde trata el uso de la información de trazabilidad en el contexto de las cadenas de transformación de modelos. Una cadena de transformación o la composición de transformación son una red de muchas sub-transformaciones, donde cada una contribuye a un objetivo de transformación más grande. Cada una de estas produce trazas que contribuyen a un grafo de trazabilidad total, el cual interrelaciona modelos dispares que representan aspectos diferentes de un sistema (p.ej funcional, seguridad y modelo de persistencia) producidos como la parte de una cadena de transformación. En algunos casos es importante conocer con exactitud como estos aspectos están relacionados a fin de realizar transformaciones adicionales (p.ej qué base de datos almacena un atributo particular del modelo funcional); por lo general esta información es escondida en los modelos de trazabilidad. El modelo propuesto permite la generación de trazas intra-modelos y entre-modelos. No obstante, el enfoque no especifica muchas de los ítems que enuncia. No describe un mecanismo para la verificación de la consistencia y la completitud, pero las cadenas de transformación podrían garantizar estos atributos en los modelos destino [Vanhooff et al. 2007].

Construyendo clasificaciones de trazabilidad en la ingeniería orientada a modelos. Paige et al. presentan un proceso que clasifica la trazabilidad de forma iterativa. El objetivo de este enfoque es levantar y analizar relaciones de trazabilidad con el fin de determinar cómo estas encajan en una clasificación de trazabilidad. Así, se busca mejorar el entendimiento de los elementos involucrados en las relaciones tales entre sus artefactos, semántica, y su dominio de aplicación. La consistencia se maneja por medio del vínculo modelo-modelo el cual se divide en dos tipos [Paige et al. 2008]:

- Estáticos, que representan relaciones estructurales que no se cambian con el tiempo. Por ejemplo: vínculos consistente-con donde dos modelos deben permanecer consecuentes el uno con el otro, p.ej, un diagrama de clase y secuencia; vínculo dependencia donde la estructura y el sentido de un modelo dependen de un segundo (subdivisión: is-a, has-a, part-of, import, export, usage, y refinement).
- Dinámicos, los cuales representan información de los modelos que puede cambiar en el tiempo. Estos incluyen los siguientes vínculos: `call`, `notifies`, `generate`, `runtime`, y `synchronized-with`.
- Otros vínculos que aportan a la verificación de la completitud son:
    - `satisfies` el cual indica qué propiedades o requisitos capturados en un artefacto son satisfechos por un modelo (puede variar en verifies o certifies).

- `Allocated-to` el cual es usado cuando un artefacto no modelo es asignado a un modelo específico que representa la información.
- `Performs` el cual indica que una tarea descrita en un artefacto es realizada por un modelo específico.
- `Explains` y `Support` indican que un modelo es explicado por un artefacto no modelo tal como documentación en lenguaje natural.

Aplicación de la transformación de modelos en el marco genérico para trazabilidad. Derezińska and Zawłocki proponen un framework que soporta la trazabilidad de diseños orientados a objetos. El framework está orientado a resolver problemas de identificación de dependencias e inconsistencias en un proyecto, analiza el impacto del cambio en un proyecto, soporta el entendimiento de los modelos incluyendo ingeniería inversa y sistemas legados, crea documentación de un proyecto, y da soporte a modelos y código. El análisis de trazabilidad se realiza en un modelo de Proyecto según un subconjunto de reglas de trazabilidad. Cada regla se define en un autómata de estado finito el cual es interpretado por un motor del marco de trabajo llamado analizador de trazabilidad. Aunque el marco no incluye un mecanismo de verificación de consistencia y completitud de los modelos, la forma como se ejecutan las reglas de trazabilidad pueden garantizar un grado de consistencia entre los modelos [Derezinska and Zawłocki 2008].

Análisis de transversalidad en la transformación de modelos. Berg et al., presentan un marco de trabajo para analizar el impacto de cambio en caso de que se presenten relaciones transversales "crosscuting" en las transformaciones de modelos. El "crosscutting" ocurre cuando en un mapeo entre fuente y destino trazan un mapa de un elemento de la fuente a dos o más elementos destino y al menos uno de estos elementos destino tiene un mapeo de uno de esos elementos de la fuente.

El análisis de impacto está basado en la trazabilidad de las dependencias entre elementos en artefactos de software. Define los intereses transversales basados en un patrón de trazabilidad (TPC - *Traceability Pattern Crosscutting*) y las relaciones de dependencia entre elementos de la fuente y el destino las cuales proporcionan la trazabilidad entre ellos así como las relaciones intra- y entre-niveles (aunque el intra- no se desarrolla en esta propuesta, este denota acoplamiento entre elementos de un nivel de abstracción específico). El análisis de impacto de dependencias transversales se realiza por medio de reglas de transformación al nivel de metamodelo y modelo [Berg et al 2006b].

Gestión del cambio basada en trazabilidad en correlaciones operacionales. Kurtev et al. complementan el trabajo de Berg [Berg et al 2006b] asociando cada traza a una regla de

transformación. La traza se define en función del elemento fuente, el elemento destino y la regla de transformación. Especifica la dependencia de traza inter-nivel (entre modelos de diferente nivel de abstracción). El cambio es definido en función del tipo de cambio, el modelo, y los elementos impactados para crear el nuevo modelo. Los cambios son incrementales y siempre son de la fuente al destino, lo cual provee un trazado hacia delante (forward). Cuando se realiza un cambio en un modelo de la fuente, se mantiene el modelo destino consistente implementando el cambio en este y sus elementos impactados. Se asume que el modelo destino no tiene cambios de forma particular.

El impacto del cambio se considera en caso de enredos, dispersiones y transversal, tanto para dependencias de dos niveles como dependencias de nivel múltiple. El impacto de cambio en el patrón de trazabilidad se operacionaliza en términos de elementos implicados en el cambio de un elemento de la fuente. El conjunto de estos elementos se llama conjunto de impacto. Kurtev define dos operaciones básicas de cambio básicas: update y delete; de forma complementaria se utiliza el comodín "*" (se especifica formalmente su uso) [Kurtev et al. 2007].

Un motor de trazabilidad dedicado a la transformación de modelos para la ingeniería de software. Amar et al. presentan un enfoque que define un motor de trazabilidad en la transformación de modelos, en un proceso dirigido por modelos. ETraceTool es la plataforma que contiene el motor y realiza la trazabilidad de las transformaciones de un modelo. Esta permite generar todos los vínculos de trazabilidad durante una transformación. No presenta una propuesta para la verificación de la completitud y la consistencia. Las principales características de la plataforma son [Amar et al. 2008]:

- El código de generación de la traza no es intruso en el código de transformación.
- La generación de la traza es explícitamente activada por el diseñador de la transformación.
- Los modelos de traza son aislados de la fuente y apuntan modelos implicados en la transformación.
- Los modelos de traza pueden ser usados en niveles de diferente granularidad.

Hacia la rigurosidad de la trazabilidad modelo a modelo. Drivalos et al. proponen la captura y especificación una rica semántica de vínculos trazabilidad entre los modelos involucrados en una transformación. Para lograr esto crean un metamodelo que permite definir los vínculos de trazabilidad así como las restricciones entre los modelos para verificar y validar los vínculos establecidos. Este enfoque no presenta un mecanismo de verificación explícito de consistencia y completitud de los modelos [Drivalos et al. 2008].

Otros enfoques orientados por aspectos presentan propuestas orientadas a la transformación de modelos en el desarrollo dirigido por modelos. Pero estos no contribuyen directamente con especificaciones de trazabilidad; ellos orientan sus propuestas de transformación desde PIM (*Platform Independent Model*) hacia PSM (*Platform Specific Model*) y determinan la transformación de modelos a partir de reglas de transformación que conservan la separación de asuntos y su composición [Kulkarni and Reddy 2003], [Solberg et al. 2005], [Reddy et al. 2006].

2.5.4 La Trazabilidad en la Industria del Software

Los enfoques evaluados en las secciones anteriores proporcionan diferentes mecanismos para realizar la trazabilidad. Pero la práctica de la trazabilidad en las empresas de desarrollo es otra realidad. En la mayoría de ellas no es una práctica estandarizada y depende cien por ciento de la metodología, los criterios de calidad definidos para el proceso de desarrollo, y las buenas prácticas de chequeo y validación que utilicen los equipos de desarrollo.

Para identificar cómo se realiza la práctica de trazabilidad, su nivel de adopción y uso orientado a la validación y verificación de la consistencia y completitud de los modelos, se realizó un estudio en ambientes reales de desarrollo de cinco empresas desarrolladoras de software de la ciudad de Medellín (Colombia) certificadas en el nivel 3 de CMMI. El mecanismo utilizado para recolectar la información fue la encuesta a través de cuestionario (definido en el Apéndice A) y se apoyó con algunas entrevistas personales.

A partir de la información recolectada se generan las siguientes conclusiones:

Las empresas en general no trabajan con un modelo conceptual o metamodelo que guíe la práctica de la trazabilidad. El lenguaje de modelado de los requisitos y artefactos es UML y las metodologías de desarrollo de software que usan el modelo iterativo e incremental el cual facilita la trazabilidad al realizar prototipos y retroalimentaciones constantes con el usuario. Algunas empresas usan la información proporcionada por los artefactos RUP (*Rational Unified Process*) [Arlow and Neustandt 2005].

La técnica de trazado más utilizada son las matrices de trazabilidad debido a que son soportadas por las herramientas de modelado CASE aunque muchas empresas desconocen o no hacen uso de los modelos de trazabilidad. Sin embargo, las empresas manifiestan la importancia de seguir un modelo o método para conocer el impacto de los cambios, tomar acciones correctivas sobre los requisitos y su modelado durante todo el ciclo de vida de desarrollo. Así, será posible minimizar las desviaciones detectadas en el proceso y asegurar la correcta interpretación de los

requisitos por parte de todo el equipo de trabajo. Además, se podrá garantizar la calidad del producto final, entendiendo esto como la plena satisfacción de las necesidades del usuario a partir de los requisitos identificados, y hacer seguimiento al cumplimiento de estos en función de características tales como definición, evolución, completitud e integración.

En general, las empresas expresan que la trazabilidad se puede facilitar si esta comienza desde el inicio del ciclo de vida. Esto significa hacer un modelado detallado de los requisitos que permita:

- Clasificarlos tempranamente
- Realizar el seguimiento y control de su proceso de transformación
- Incluir actividades de trazabilidad previamente establecidas
- Estar apoyados en modelos construidos en UML, u otros lenguajes de modelación sistemáticas, y herramientas CASE que provean matrices de trazabilidad
- Usar repositorios centralizados de almacenamiento para los diferentes artefactos los cuales facilitan notoriamente la realización de la trazabilidad permitiendo que los diferentes roles puedan acceder a información reciente o actualizada de forma sencilla y rápida.

Entre las actividades que generalmente usan las empresas para verificar la dependencia de requisitos y artefactos están: reuniones de obtención de requisitos donde el grupo de desarrollo verifica la relación entre los diferentes participantes, sus requisitos y el impacto del cambio para los diferentes requisitos en los cuales se observa interdependencia. En la etapa de diseño, se verifica la dependencia utilizando casos de uso y escenarios de prueba. En la fase de transición o pruebas se valida su evolución hasta la implantación y monitoreo.

Para localizar y verificar la evolución de un requisito en cualquier etapa del ciclo de vida se deben tener diferentes puntos de chequeo (o listas de chequeo) en los cuales los miembros del equipo en forma individual hacen validaciones de la evolución de los requisitos (utilizando prototipos y entregables de cada etapa) y luego se hace retroalimentación para verificar en conjunto los resultados de la validación. De esta forma, todos revisan que el avance del requisito en la etapa sí ha sido ejecutado de acuerdo a los lineamientos iniciales.

El control del cambio se hace principalmente por medio de sistemas de versiones y plantillas de control. La transformación es un proceso intuitivo que depende de la experticia de los analistas y desarrolladores. Algunos se apoyan en las herramientas de desarrollo que comúnmente proveen transformación del modelo Entidad-Relación [Chen 1985] al Modelo Relacional [Codd

1970] y de los diagramas de comportamiento del diseño a código. La trazabilidad no la pueden obtener a partir de reportes generados por medio de la herramienta que describan en lo posible una matriz CRUD.

Debido a las anteriores y otras razones es común ver que la mayoría de las empresas de desarrollo de gama media certificadas en el nivel 2 ó 3 de CMMI aún no logran implantar la práctica de la trazabilidad de requisitos de forma integral al proceso de desarrollo.

Además, el cuestionario proporcionó información complementaria acerca de otros conceptos adicionales importantes que permiten analizar las condiciones actuales de entendimiento, adopción y uso de la práctica de la trazabilidad. Las empresas tienen conciencia de la necesidad de tener en cuenta situaciones que pueden entorpecer la práctica de trazabilidad, algunas de ellas son:

Recurso humano no disponible. Es posible encontrar participantes reacios a levantar la documentación requerida. También, roles mal definidos en el grupo de trabajo no permite la especialización del conocimiento en lo que respecta a los artefactos creados en diferentes etapas del ciclo de vida. Muchas veces no existe un responsable del equipo de trabajo que valide la evolución de los requisitos y haga un buen uso de las herramientas de desarrollo con respecto a la trazabilidad.

Alcance mal definido. La trazabilidad se debe realizar a través de todo el ciclo de vida de desarrollo, pero muchas veces no se considera necesaria durante el levantamiento del dominio del problema o en la implantación del sistema. Si el analista no tiene claro cual va a ser el origen sobre el cual se debe partir y hasta donde se debe seguir la traza o viéndolo de otra forma qué tanto debe trazar, será difícil asegurar qué las actividades de trazado se realicen. De igual forma, medir el impacto del cambio estaría mal dimensionado ya que la mayoría de las empresas no tienen un método establecido o dependen de la experiencia de los analistas o líderes de los proyectos.

Clasificación incorrecta de los requisitos. Aunque la clasificación de los requisitos puede verse como una actividad muy simple, la mayoría de veces en las empresas dependen de la experiencia de los analistas de requisitos. Normalmente sólo separan requisitos funcionales, no funcionales, otros como requisitos de información y eventualmente los transversales.

Las empresas expresaron la importancia de hacer la separación de requisitos ya que son indispensables para la toma de decisiones de arquitectura, la definición de las calidades sistémicas y el esbozo de la matriz tecnológica del producto. Las empresas aún no conocen la técnica de identificación de asuntos transversales. Los procesos y documentación referenciada

transversalmente por otros procesos es usadas por las empresas eventualmente para no perder la interdependencia entre requisitos.

Desconocimiento de los beneficios que la trazabilidad. Esta situación sigue siendo uno de los problemas más comunes. No utilizar puntos de chequeo estandarizados a través del ciclo de desarrollo impide saber cuál ha sido la evolución de los requisitos, la transformación de los modelos y la forma como la trazabilidad se ha venido realizando.

Ante la falencia del uso analítico de las matrices de trazado o de otros modelos, métodos o técnicas de trazabilidad a lo largo del ciclo de vida de desarrollo, las empresas consideran necesario atender este problema para garantiza un producto final de alta calidad validado durante todo el tiempo. Además, esto les permitirá tener un completo conocimiento de los requisitos en todo el equipo de trabajo lo cual asegura que se esta apuntando hacia un mismo objetivo y no existan posibles desviaciones por falta de conocimiento. Finalmente se podrá asegurar una cobertura total de los requisitos en el producto final ya que se tiene un alcance previamente definido y conocido por todo el equipo lo cual sirve como fuente de consulta y chequeo en cada etapa.

Pocas empresas usan los modelos de trazabilidad construidos desde las herramientas CASE. Para soportar esta práctica tienen descritos procedimientos para comunicar a todos los participantes directamente involucrados en el proceso cuándo y de qué manera deben realizar cambios en los modelos de desarrollo pero no en los modelos de trazabilidad.

El costo asociado a la trazabilidad. Este fue uno de los aspectos de gran relevancia para las empresas. Estas exponen que el trazado requiere una dedicación de recursos (personal, equipos y tiempo) en actividades propias de la práctica durante el proceso de desarrollo. Muchas veces en organizaciones pequeñas es difícil contar con personal disponible y dedicado para las actividades de trazado en comparación con las grandes empresas que les es más fácil asignar recursos a actividades especializadas, pero el hecho de no contar con más empleados exige organizarse y definir puntualmente quién hace qué actividades, en qué momento y de qué forma.

Métricas de trazabilidad. Las empresas no utilizan métricas para esta práctica. La inclusión de estas se tiene considerada como una de las mejoras que se quiere dar al proceso de desarrollo de software. La utilización de documentos de trazabilidad ya sean físicos o electrónicos permiten registrar relaciones de trazado de gran importancia; estos serán la carta de navegación y de apoyo sobre los cuales se realicen las medidas correspondientes y el

mantenimiento de la aplicación. Entre los documentos más comunes se encuentran: especificación inicial de requisitos (documento de visión), análisis de factibilidad, documentos de diseño y modelado de requisitos, diccionario de datos, funcionalidades implementadas, casos de prueba basados en requisitos validados.

2.6 Discusión acerca del Tratamiento de la Trazabilidad

Este capítulo hace una revisión del estado de la práctica de la trazabilidad desde diferentes contextos del desarrollo de software con el fin de conocer y entender la forma como diferentes enfoques logran consistencia y completitud a partir de la trazabilidad. La evaluación se hizo desde los modelos tradicionales orientados a la trazabilidad, el desarrollo orientado a aspectos, el desarrollo dirigido por modelos y el contexto de aplicación de la industria del desarrollo de software. Aunque parece un amplio espectro de revisión, esto permitió visualizar algunos asuntos que hacen de la trazabilidad una práctica vulnerable a su aplicación y al valor agregado que provee para los equipos de desarrollo.

La trazabilidad es una práctica que se debe realizar desde los requisitos hasta la implementación. Sin embargo, como se observó en la evaluación, su uso está orientado a realizarse en las etapas iniciales o al final del proceso de desarrollo. Pero en los modelos intermedios que se generan en la fase de análisis y parte del diseño, la trazabilidad pierde vigencia haciendo que las matrices o los modelos de trazabilidad dejen de ser usados, los modelos de desarrollo cambien y las trazas temporales se haga directamente en los modelos de desarrollo. Entonces la verificación de la consistencia y la completitud no se hacen y por lo tanto no es fácil hacer mediciones al respecto. Esto aumenta el riesgo durante el proceso de desarrollo y una vez los productos sean puestos en marcha.

Los modelos de trazabilidad propuestos desde los diferentes puntos de vista y enfoques son alternativas pertinentes que se pueden adoptar en la industria del software. No obstante, si estos no están soportados por una herramienta de modelado o estrategia de desarrollo que se integre directamente con los atributos de calidad, su adopción y uso se dificulta, debido al bajo nivel de verificación de la calidad del software que se realiza durante el proceso de desarrollo. A esto se suma la falta de recursos humanos y de tiempo que disponen los equipos de desarrollo para la entrega de sus productos de software, el alcance de diferentes tipos de proyectos, y el nivel de confiabilidad y exactitud con la que se entregan los productos.

Muchos de los enfoques usan XML como lenguaje para la especificación de los requisitos, definición y registro de la traza. Esto permite orientar la implementación de un nuevo enfoque de

trazabilidad usando el lenguaje scripting para definición, búsquedas y análisis de la trazabilidad y el impacto del cambio que facilite la generación de modelos destino de una transformación con modelos exportados en XMI de las herramientas de modelado CASE. Así, nuevos enfoques podrían ser adoptados gradualmente por la industria del software.

De los contextos revisados se seleccionaron los enfoques con trabajos más robustos que pueden soportar directa o indirectamente la verificación de la consistencia y la completitud desde los modelos de trazabilidad. La Tabla 1 resume las características más importantes que hacen posible esta verificación.

Los enfoques de Egyed representan el trabajo con mayor madurez y evolución en el tiempo donde completitud y consistencia están asociadas a la estrategia de trazabilidad aplicada, por esta razón el modelo definido en esta disertación será comparado con este. En los contextos de la orientación a aspectos y el desarrollo dirigido por modelos aún requiere más trabajo para lograr estos atributos de calidad o al menos mostrar mayor experiencia en este campo.

En los modelos orientados a aspectos, el enfoque de Berg et al. es la propuesta más concreta del manejo de la trazabilidad pero no se evidencia el tratamiento de los atributos de calidad. De igual forma, en los modelos dirigidos por modelos, el enfoque de Amar et al. puede orientar una verificación de los atributos de calidad pero no presenta una propuesta concreta al respecto.

Los enfoques AOSD proveen diferentes elementos trazables, pero no vínculos de trazado o métodos de validación y verificación. Los enfoques no están orientados a la trazabilidad, se centran en el tratamiento de los asuntos transversales y no en la forma como podrían proporcionar consistencia y completitud a los modelos de desarrollo, el manejo del cambio y la transformación de los asuntos transversales. Sin embargo, la separación de asuntos y su composición puede dirigir un enfoque orientado la trazabilidad que facilite la verificación de los atributos de calidad.

Propuestas recientes, como la del proyecto Europeo AMPLE[13], soportan la trazabilidad por medio de un marco de trabajo que provee funciones orientadas al impacto del cambio [Galvão et al. 2008]. Particularmente, en este proyecto se modela e implementa la trazabilidad de la variabilidad de los modelos y artefactos SPL (*Software Product Line*). El marco de trabajo proporciona servicios de trazabilidad para recuperar y buscar vínculos de traza entre artefactos específicos. El objetivo específico es crear búsquedas especializadas de trazado tales como requisito/característica cubierta, análisis del impacto del cambio, trazado de variantes de producto, entre otros [Galvão et al. 2008].

[13] www.ample-project.net

Tabla 1. Comparativo de los enfoques más representativos en cada contexto con respecto al soporte a la Consistencia y la Completitud de requisitos y modelos.

Contexto	Enfoque	Características de Verificación de Consistencia & Completitud			
		Consistencia	Completitud	Mecanismo	Herramienta
Modelos orientados a la Trazabilidad	Egyed (2003, 2004, 2005, 2006)	Estrategia 1: definición de trazas hipotéticas excesivas. Estrategia 2: reglas de Consistencia estáticas y determinísticas.	Estrategia 1: definición de trazas hipotéticas excesivas. Estrategia 2: reglas de Consistencia estáticas y determinísticas.	Estrategia 1: si un elemento del modelo m es exactamente alguna huella de trazado f (en el código), entonces se reconoce que m traza a dicha huella de trazado y no a otra. Estrategia 2: El alcance de una regla de consistencia está completo al contener subcondiciones AND/OR que influencian cómo y si los elementos de modelo son accedidos.	UML/Analyzer integrada al IBM Rational Rose™
Modelos Orientados a Aspectos	Tekinerdogan (2007)	Cuando hay un cambio, las vistas arquitectónicas necesitan ser sincronizadas para preservar la consistencia entre las vistas.	NA	NA	M-Trace, herramienta que usa representaciones basadas en XML.
Modelos Dirigidos por Modelos	Paige (2007)	Vínculos de traza: consistent-with	Vínculos de traza: Satisfice, Allocated-to, Performs	Definición y soporte a los vínculos de trazabilidad	NA

2.7 Escenarios de Trabajo para la Solución

De acuerdo al estudio realizado, es importante desarrollar el objetivo de esta disertación desde cuatro escenarios de trabajo que facilitarán la demostración de la tesis y darán soporte a las debilidades encontradas en el tratamiento de la trazabilidad. Cada escenario se describe a partir de un conjunto de premisas que facilitan la definición de requisitos básicos que deben ser atendidos para establecer y mantener la traza de los requisitos y artefactos durante el proceso de desarrollo. Los escenarios están orientados a soportar y controlar la trazabilidad de forma integral y estándar en el proceso de desarrollo para lograr consistencia y completitud. Además serán la guía para describir el patrón de trazabilidad objetivo de esta disertación.

2.7.1 Escenario 1: Modelos de Trazabilidad

Este escenario se define a partir de las siguientes premisas y requisitos.

Premisa 1. La trazabilidad es una práctica que depende generalmente de las metodologías de desarrollo y decisiones de calidad que utilizan los analistas y desarrolladores. Este escenario muestra la necesidad de hacer de la trazabilidad un patrón que controle la evolución de los requisitos durante el proceso de desarrollo.

Premisa 2. Aunque las matrices son el mecanismo más usado para soportar la trazabilidad, su actualización depende del tipo de relación usada por el grupo de desarrollo y los elementos que deseen trazar. Esto hace vulnerable el registro de la trazabilidad y su uso para soportar las funciones de valor agregado.

Premisa 3. Los requisitos evolucionan pero los elementos de modelo correspondientes en otros niveles de abstracción no cambian consistentemente. Durante el proceso de desarrollo es común ver que la mutabilidad de las reglas del negocio, y los cambios en las decisiones tecnológicas que soportan la arquitectura del sistema, cambian el sentido de los requisitos iniciales, o simplemente provoca incompletitud e inconsistencias entre los elementos de modelo en el mismo o diferente nivel de abstracción.

Premisa 4. La traza generalmente se realiza hacia delante y horizontal, dejando la traza vertical y hacia atrás un poco relegadas por falta de un mecanismo que provea vínculos de trazado intra-nivel o facilite una transformación asociada a la ingeniería inversa.

Estas premisas conducen la formulación de los siguientes requisitos para lograr el objetivo de la disertación:

1. Definir un metamodelo que provea los elementos y el mecanismo necesario para estandarizar la práctica de trazabilidad desde criterios de calidad, tipo de proyecto y políticas más usadas en la industria del software o definidos por equipos de desarrollo en diferentes proyectos.
2. Facilitar la definición de elementos trazables y vínculos de trazado para un modelo de trazabilidad que estandarice y guíe la práctica de la trazabilidad como parte integral del modelado de los requisitos y artefactos durante todo el proceso de desarrollo.
3. Los elementos trazables y sus relaciones deben ser controlados por dirección (hacia delante, hacia atrás) y forma de trazabilidad (horizontal, vertical), para garantizar un mecanismo de verificación de consistencia y completitud de los modelos.
4. Los modelos de trazabilidad deben proporcionar un mecanismo que facilite la traza bidimensional o multidimensional.
5. Proporcionar un mecanismo de transformación que garantice el refinamiento y mapeo de los elementos de modelo trazables en diferentes niveles de abstracción, manteniendo la consistencia y completitud de los modelos.
6. Identificar elementos que pueden ser parte de los modelos de la fuente y los modelos del destino de una transformación que deben participar en la trazabilidad de dichos modelos.
7. Definir un mecanismo para crear y mantener reglas de transformación. Su definición y uso se debe realizar a partir de los elementos definidos por el patrón de trazabilidad.
8. Planear la ejecución de las reglas de transformación de tal forma que faciliten servicios de trazabilidad tales como la consistencia y la completitud y el manejo del cambio.
9. Formalizar las reglas en un lenguaje de transformación declarativo para facilitar la operacionalización de la trazabilidad.

2.7.2 Escenario 2: Descomposición y composición de modelos

Este escenario se define a partir de la siguiente premisa y requisitos.

Premisa 5. Cuando el problema es complejo consecuentemente los modelos de desarrollo de la solución tienden a ser complejos, independiente del estilo de modelado, separación de funciones u otras estrategias para minimizar el impacto del esfuerzo durante el desarrollo. La trazabilidad se ve afectada por esa situación, convirtiéndose en una práctica costosa y tediosa que los analistas descartan con facilidad. Además, cualquier cambio tiene alta probabilidad de ser invasivo (al

dispersarse o enredadarse en diferentes elementos) debido al grado de complejidad de los modelos. Este escenario muestra la necesidad de proporcionar un marco de trabajo que facilite el modelado de los requisitos por medio de la separación de diferentes tipos de asuntos identificables en el sistema. El nivel de complejidad del sistema se debe controlar por medio de los asuntos identificados y su evolución debe conservar la descomposición y composición de forma estándar de acuerdo a los recursos que los analistas usan en su proceso de desarrollo.

Esta premisa lleva a formular los siguientes requisitos.

10. Establecer un mecanismo de identificación y descomposición de asuntos y requisitos del usuario y del sistema que facilite la trazabilidad desde las primeras etapas del desarrollo. Esto permitirá establecer de forma estándar los elementos trazables y sus relaciones en un patrón de trazabilidad.
11. Definir relaciones de dependencia y composición entre los asuntos.
12. Controlar el impacto del cambio y su propagación en diferentes asuntos del sistema desde los Modelos de Trazado.

2.7.3 Escenario 3: Estimación y Propagación del Cambio

Este escenario se define a partir de la siguiente premisa y requisitos.

Premisa 6. Los cambios se pueden gestionar de diferentes formas, siendo más efectivo desde el registro de la traza. No obstante, el registro de la evolución de los requisitos y sus cambios, no se realiza estrictamente o se limita a plantillas estandarizadas para la negociación de cambios. Esto no permite ver al grupo de desarrollo el nivel de penetración (en algunos casos invasión) que un cambio ocasiona en diferentes modelos. En consecuencia no sabe qué tan complejo es el cambio o que tan enredados deja este a los modelos. Este escenario muestra la necesidad de proporcionar el manejo y la propagación del cambio a partir de los modelos de trazabilidad.

Esta premisa lleva a formular los siguientes requisitos.

13. Definir un patrón de trazabilidad que garantice el uso dinámico de las reglas de transformación de acuerdo a las necesidades provocadas por la evolución o los cambios en los requisitos, y presentar alternativas de transformación cuando sea necesario.
14. Establecer los eventos que los usuarios y grupo de desarrolladores podrían ocasionar un cambio en modelos existentes.

15. Todo el evento de cambio debe ser generado a partir de un requisito nuevo o la modificación de uno ya existente en un modelo fuente. Cada evento de cambio debe estar asociado mínimo a una operación de cambio.
16. Determinar el procedimiento de ejecución de cada una de las operaciones de cambio: crear, actualizar y eliminar.
17. Estimar el impacto del cambio a partir del modelo de trazabilidad definido para el sistema en desarrollo.
18. Propagar el cambio a partir del modelo de trazabilidad definido para el sistema en desarrollo.

2.7.4 Escenario 4: Marco Metodológico

Los procesos o metodologías de desarrollo actuales establecen actividades, recursos, y entregables que orientan la evolución y el desarrollo de los requisitos hacia productos de software. Generalmente, las metodologías se orientan a generar elementos de desarrollo y control por cada etapa del ciclo de vida de tal forma que un proyecto cumpla y provea los productos software requeridos por el usuario. Sin embargo, aunque se habla de las "mejores prácticas" de desarrollo de software, estas metodologías no están soportadas por mecanismo de control y verificación que faciliten la confiabilidad de un producto software.

Actividades tales como la transformación, y la trazabilidad de los asuntos y requisitos del sistema, son secundarias y generalmente intuitivas. En algunos casos donde el proceso de desarrollo está formalizado y documentado (el caso de las empresas certificadas en CMMI nivel 3) las actividades de configuración del software son transversales al proceso de desarrollo. Sin embargo, algunas actividades de verificación como la consistencia y completitud de los modelos no son frecuentemente realizadas debido al costo que estas ocasionan.

Premisa 7. La formulación de un patrón de trazabilidad debe ser adoptado por medio de un marco metodológico que establezca estas actividades de trazabilidad y transformación como disciplinas que guían el desarrollo de software orientado a modelos.

Esto guía la formulación de los siguientes requisitos:

19. Identificar la forma en que actividades de transformación y trazabilidad pueden establecerse como base de un nuevo marco metodológico.

20. Establecer un marco de gestión de la configuración que soporte el desarrollo dirigido por modelos.
21. Definir actividades, recursos y entregables que interactúan en del marco metodológico.

2.8 Conclusiones

En este capítulo se hizo una revisión del estado de la práctica de la trazabilidad como medio para lograr la verificación de la consistencia y completitud de los requisitos a través de los modelos de desarrollo en diferentes niveles de abstracción. Para lograr esto, el capítulo se divide en dos partes fundamentales: en la primera parte se definen los conceptos generales que definen la trazabilidad y su tratamiento. Las definiciones facilitan el entendimiento de los diferentes escenarios discutidos en esta disertación. En la segunda parte se realizó una evaluación de la trazabilidad desde tres contextos de la ingeniería del software: los modelos orientados a la trazabilidad, los modelos del desarrollo de software orientado por aspectos, y los modelos del desarrollo dirigido por modelos. Para complementar la evaluación se hizo una revisión del tratamiento de la trazabilidad en las empresas de desarrollo de software.

El estudio proporcionó la información necesaria para identificar el nivel de investigación alrededor de la trazabilidad y las oportunidades de crear un nuevo método o técnica de trazabilidad que combine la visión y estrategias de desarrollo proporcionadas por los nuevos paradigmas de desarrollo AOSD y MDD. Esto facilitará el abordaje de esta práctica con el objetivo de obtener la información necesaria para realizar funciones de verificación de calidad del software, en especial lo que corresponde a la consistencia y la completitud.

De acuerdo al estudio realizado en las empresas de desarrollo y de servicios, muchos desarrolladores piensan que la trazabilidad es un proceso costoso que no trae muchos beneficios, pero también encontramos otros que la usan como la herramienta base para estimar el valor de los cambios solicitados por los participantes durante el proceso de desarrollo o después de la puesta en marcha de los productos de software. Además, ayuda a disminuir los conflictos que comúnmente se presentan entre los participantes.

Aunque muchos de los modelos de trazabilidad propuestos en diferentes enfoques proveen semántica e información suficiente para soportar el trazado de los requisitos, la mayor debilidad se reafirma en cuanto a la adopción de los modelos de trazabilidad y el uso de la información que provee la traza. Por esta razón, otros atributos de calidad como la consistencia y la completitud son

tratados por muy pocos enfoques y sin la fortaleza necesaria para que los analistas vean en la trazabilidad la práctica necesaria para lograr productos confiables.

Es necesario también minimizar el grado de separación entre las prácticas de modelado y la trazabilidad. Durante el proceso de desarrollo, modelar los requisitos en artefactos de la solución que representan el problema y llevar su trazabilidad, son tareas independientes. Esto exime al grupo de desarrollo de las tareas de calidad que ayuda a minimizar conflictos entre los participantes, orienta hacia una correcta valoración del modelado, y determina el impacto del cambio.

Lograr desarrollar los requisitos establecidos en los escenarios de solución permitirá hacer de la trazabilidad una práctica más allá de un simple registro histórico de la evolución de los requisitos. Con esta tesis se pretende dar un sentido práctico, necesario e intuitivo que genere valor agregado al proceso de desarrollo en aspectos tales como: (i) el trazado de artefactos de acuerdo a su función en la descomposición y composición de modelos y entre elementos de modelos de un mismo nivel y de diferente nivel; (ii) la automatización del trazado como parte del modelado de los requisitos y sus artefactos durante el proceso de desarrollo; (iii) el control del proceso de cambio desde la transformación de modelos; (iv) la generación de información para realizar correctivos tempranos y dar soluciones de bajo costo para evitar conflictos futuros entre los participantes del proyecto de desarrollo.