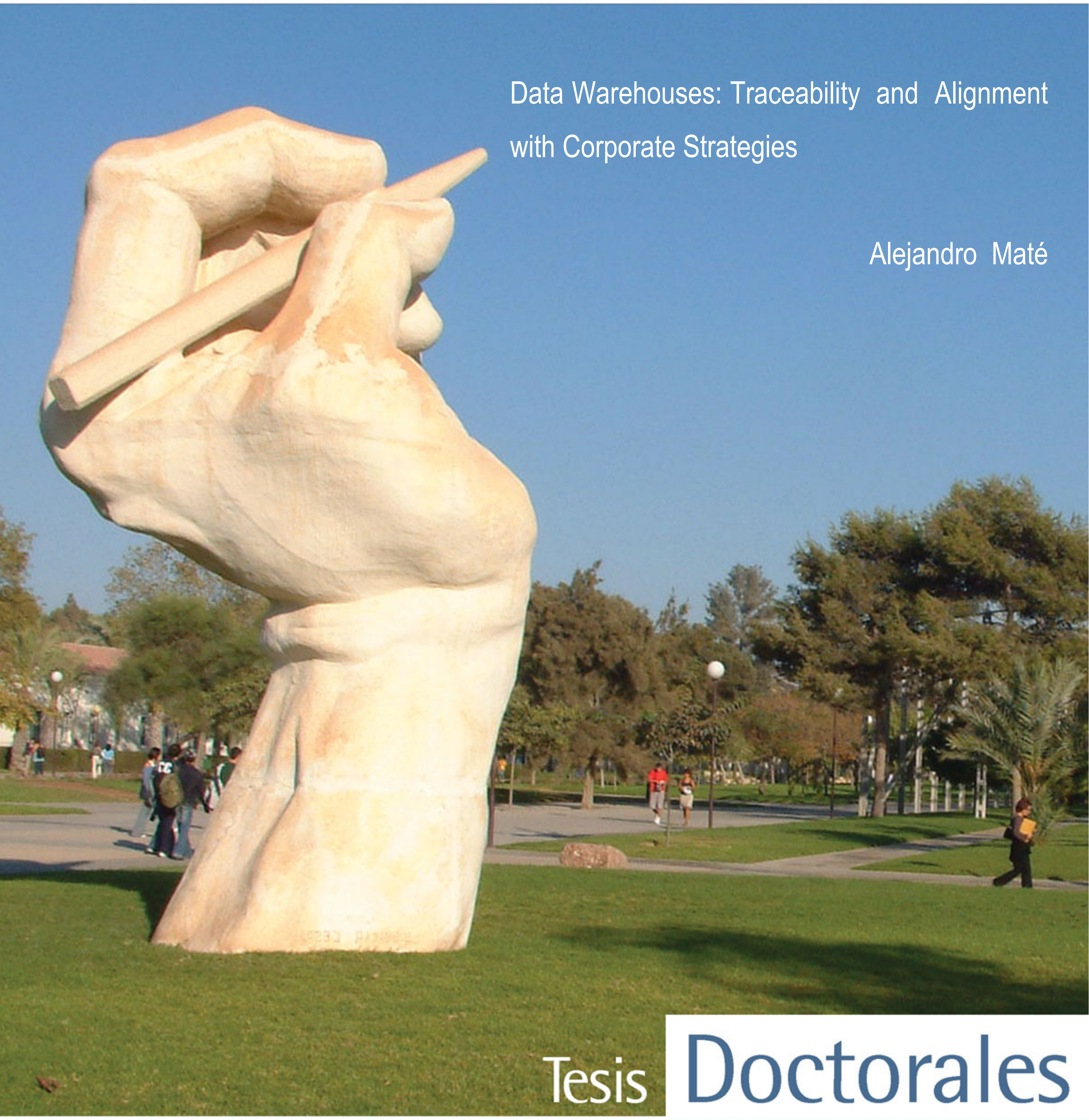


Data Warehouses: Traceability and Alignment with Corporate Strategies

Alejandro Maté

Alejandro Maté Morga

Data Warehouses: Traceability and Alignment with Corporate Strategies

- Ph.D Thesis -

Advisor: Juan Carlos Trujillo Mondéjar

Lucentia Research Group
Departamento de Lenguajes y Sistemas informáticos
Universidad de Alicante

"La ciencia se compone de errores, que a su vez, son los pasos hacia la verdad."
Julio Verne

Agradecimientos

Esta Tesis Doctoral es el resultado conjunto del duro trabajo y esfuerzo no sólo mío, sino también de mi familia, profesores y amigos que han realizado durante mi vida para apoyarme.

A mi familia, por cuidar de mi cuando era joven y permitirme ir a la universidad.

A mis profesores, tanto en el instituto como en la universidad, por enseñarme todo lo que necesitaba y cómo pensar por mi mismo.

A mis amigos, que han estado apoyandome durante mi vida. En especial a Patricia, por escuchar mis constantes quejas, y a Xavier y a Andrés por apoyarme durante mis años de universidad.

A mi director de tesis, Juan Carlos, por su consejo y apoyo, y por enseñarme como escribir de forma que tuviera sentido.

A mis compañeros en el DLSI, Victor, Miquel, Xavi, Isabel, Jose Alfonso, y Elisa.

Y a Abayomi por empujarme a escribir esta Tesis Doctoral.

Acknowledgements

This PhD thesis is the result of the combined hard work and effort that not only me, but also my family, teachers and friends have put during my life to support me.

To my family, for taking care of me when I was young and allowing me to go to the university.

To my teachers, both in high school and the university, for teaching me everything I needed and how to think by myself.

To my friends who have been supporting me during my life. Especially to Patricia for hearing my constant rambling, Xavier and Andrés for supporting me during my university years.

To my advisor, Juan Carlos, for his advice, guidance, and teaching me how to write making sense.

To my partners at DLSI, Victor, Miquel, Xavi, Isabel, Jose Alfonso, and Elisa.

And to Abayomi for pushing me to write this PhD Thesis.

Preface

In the early 90's data warehousing techniques are proposed to store massive, historical data of the organization. The purpose of data warehouse techniques is to support the decision making process, allowing decision makers to take better informed decisions and improve business performance. In practice, building a data warehouse is an extremely costly development, that requires to carefully elicitate and model the information required by users in order to take decisions.

As data warehouses are at the core of most Business Intelligence systems, both industry and academia have put much effort into improving data warehouse development. However, so far they have only been partially successful. Currently, data warehouse projects still present a failure rate of over 70%, and a number of problems exist that have not been addressed. First, current approaches do not provide any mechanism to preserve traceability. Thus, as data warehouse multidimensional schemata are modified according to the available data, the capability of identifying the status of each requirement is lost. Second, current approaches mostly focus on designing the data warehouse repository. However, data warehouses are long term projects, thus they require adequate support for maintaining the repository and introducing changes as the business evolves. Third, data warehouses are used by several decision makers, each expert in a different business area, and rarely knowledgeable in IT. Thus, a validation process is required to ensure that requirements have been correctly gathered and the data warehouse supports the business goals to be improved.

Therefore, the present PhD Thesis tries to address these issues by proposing (i) a traceability approach for maintaining traceability throughout the data warehouse development process, (ii) a formalization to model the reconciliation process, allowing the designer to document the data source integration process and add new data sources in an easier way, (iii) a set of modules to partition requirements models and improve the communication with users, and (iv) an alignment process to ensure that the data warehouse supports the current business goals. Overall, these techniques allow us to reduce the cost and effort of data warehouse projects and guarantee their alignment with business objectives.

This PhD Thesis has been funded by the program Vali+D from the Generalitat Valenciana under the grant ACIF/2010/298, and partially supported by the MESOLAP (TIN2010-14860), SERENIDAD (PEII-11-0327-7035) and GEODAS-BI (TIN2012-37493-C03-03) projects from the Spanish Ministry of Education and Competitivity and the Junta de Comunidades de Castilla La Mancha respectively.

Alicante, October 2013

Alejandro Maté Morga

Contents

List of Figures

Part I

Summary

1

Síntesis en Castellano

Esta Tesis se presenta como un compendio de artículos publicados. Por tanto, tal y como exige la normativa de la Universidad de Alicante, este capítulo provee una descripción y un resumen de las hipótesis iniciales y los objetivos de investigación, así como el compendio de artículos que componen la presente Tesis. Para alcanzar este objetivo, este capítulo justifica e incluye un sumario del contenido científico que forma parte de cada captulo de la Tesis, incluyendo las motivaciones, las investigaciones y desarrollos realizados y las conclusiones alcanzadas.

1.1 Objetivos de Investigación e Hipótesis Iniciales

En la década de los 90 se propusieron las técnicas de almacenes de datos para almacenar cantidades masivas de datos históricos de la organización [15, 18]. El almacenamiento de datos tiene como objetivo dar soporte al proceso de toma de decisiones, obteniendo datos de diversas fuentes heterogéneas, e integrando estos datos en un repositorio para la organización. Estos datos son, posteriormente, refrescados de manera periódica con los datos más recientes disponibles de cada fuente por medio de procesos de Extracció/Transformación/Carga (ETL) [35].

El desarrollo del almacén de datos es una tarea costosa tanto en términos de recursos como de tiempo, pudiendo tardarse años en completarse dependiendo del tamaño del almacén de datos. Por tanto, un diseño cuidadoso y un mantenimiento adecuado es necesario para completar con éxito el desarrollo del almacén. Las primeras propuestas realizadas para el diseño de un almacén de datos son dirigidas por las fuentes de datos (supply-driven). Estas propuestas sólo consideran la información almacenada en las fuentes de datos [15, 18]. En las propuestas supply-driven se comienza modelando los conceptos multidimensionales, es decir, hechos y dimensiones disponibles en las fuentes de datos. Un hecho almacena medidas relacionadas con el rendimiento de un proceso de negocio (e.g. cantidad de productos vendidos) mientras que una dimensión almacena información relevante del contexto (e.g. datos de los productos). Una vez se han identificado los conceptos multidimensionales, estas aproximaciones diseñan un esquema estrella [18] a nivel lógico para crear las tablas que contendrán los datos del almacén.

Dada la ausencia de formalismos para modelar los aspectos multidimensionales de los almacenes de datos, varios trabajos se han propuesto para definir un modelo conceptual del almacén [21, 37, 10, 14, 33]. Sin embargo, aún con la introducción del modelado conceptual, las propuestas supply-driven ignoran los requisitos de los usuarios hasta que el almacén ya ha sido construido. Esto puede provocar que los usuarios queden insatisfechos y que el almacén de datos no soporte adecuadamente el proceso de toma de decisiones [9].

Para resolver este inconveniente, se proponen las aproximaciones dirigdas por requisitos (demand-driven) y las híbridas [34, 9, 25, 18, 30, 5]. Por un lado, las aproximaciones demand-driven [41, 5, 30] introducen un paso previo de análisis de requisitos en el proceso. Estas aproximaciones se centran en construir un almacén de datos prestando especial atención a los requisitos de los usuarios. A continuación, una vez se ha construido el almacén, los datos son cargados a partir de distintas fuentes.

Sin embargo, rara vez los usuarios del almacén de datos, que se componen principalmente de expertos del negocio, tienen un conocimiento detallado y preciso de las fuentes de datos operacionales de donde se extraen los datos del almacén, ya que no son expertos en tecnologa [41]. Por ello, no está garantizado (i) que toda la información requerida se encuentre efectivamente almacenada, o en el formato esperado, y (ii) que no se ha obviado información importante durante la fase de requisitos. Por otro lado, las aproximaciones híbridas [9, 25] incluyen un paso de análisis de las fuentes de datos previo a la implementación del almacén. Por ello, estas aproximaciones son capaces de identificar de forma más temprana qué requisitos no se pueden satisfacer y qué datos relevantes pueden haber sido obviados.

No obstante, incluso las aproximaciones para el desarrollo de almacenes de datos más recientes se centran en proveer de las herramientas y pasos necesarios para construir el almacén, prestando poca atención a otros aspectos, tales como:

1. La falta de un plan a largo plazo de Inteligencia de Negocio que incluya

los objetivos del negocio. Dado que los requisitos se extraen de tomadores de decisión individuales, cada uno de ellos provee de una vista parcial del problema, que puede no estar completamente alineada con los objetivos del negocio. Además, la diferencia entre el lenguaje utilizado por los tomadores de decisión y los diseñadores del almacén de datos, combinado con las vistas parciales, hacen que resulte difícil validar los requisitos de los usuarios en etapas tempranas, y ha sido identificado como una de las mayores causas de fracaso [4].

2. Asegurar que los modelos multidimensionales del almacén de datos satisfacen, efectivamente, con los requisitos especificados por los usuarios. Hasta la fecha, únicamente el trabajo presentado en [38], en el que también se basa [5], propone medidas para evalúar la calidad del esquema multidimensional del almacén de datos con respecto a los requisitos. Sin embargo, algunas de estas medidas pueden resultar difíciles de calcular, ya que requieren información de múltiples modelos, donde los mismos elementos pueden tener nombres o incluso estructura distinta y deben de ser rastreados manualmente.

3. Disminuir el esfuerzo necesario para gestionar los distintos modelos involucrados en el desarrollo de almacenes de datos. Dado que los almacenes de datos requieren la utilización de varios modelos con distintos niveles de abstracción (requisitos, conceptual, lógico and físico), rastrear un error o encontrar el stakeholder, o fuente de datos, responsable de la estructura de ciertos cubos del almacén, puede resultar un desafío.

4. Dar soporte a la evolución y mantenimiento del almacén de datos una vez ha sido implementado. Proveer de herramientas que permitan identificar el alcance de un cambio puede resultar crucial para evitar efectos no deseados en otros elementos del sistema de Inteligencia de Negocio, tales como cuadros de mando integrales y operacionales

En consecuencia, todos estos aspectos recaen en experiencia y habilidad del diseñador del almacén de datos, y aún en el caso de expertos, no se encuentra garantizado el éxito del proyecto del almacén. Por ello, las **hipótesis de esta Tesis** son que el desarrollo de almacenes de datos puede ser mejorado mediante:

1. La incorporación de trazabilidad en el proceso de desarrollo, permitiendo la automatización de la validación de requisitos con el esquema del almacén y dando soporte a la gestión de cambios conforme el almacén de datos evoluciona, acortando tiempo de desarrollo y evitando errores.

2. Permitiendo el cálculo de una serie de medidas sobre modelos complejos de almacenes de datos, que den una idea de cómo de adecuado es el modelo del almacén propuesto con respecto a los requisitos de usuario.

3. Alineando el desarrollo del almacén de datos con los objetivos de negocio que se encuentran detrás de las decisiones tomadas por los tomadores de decisión, asegurando as la construcción de un almacén de datos que soporte los objetivos de la organización.

Hasta el momento, a pesar de los esfuerzos previos, los proyectos de almacenes de datos aún presentan una tasa de fracaso que supera el 70% [4], así como una brecha entre los expertos TI y los expertos del negocio, que limita los beneficios obtenidos gracias al almacén de datos. Por ello, para desarrollar nuestra propuesta, comenzamos basándonos en una propuesta ya existente elaborada en [25] dentro del grupo de investigació Lucentia, en donde se presenta una aproximación híbrida para el desarrollo de almacenes de datos dirigida por modelos basada en la Model Driven Architecture (MDA) [19].

En conclusión, **el objetivo de investigación de esta Tesis Doctoral** es definir una serie de técnicas y aproximaciónes para mejorar el desarrollo de almacenes de datos que cubran las deficiencias existentes mediante (i) la introducción y preservación de trazabilidad en el proceso, (ii) la formalización de las relaciones involucradas en proceso de reconciliación, y (iii) el alineamiento del desarrollo del almacén con el modelo estratégico del negocio, garantizando as que el almacén soporta los objetivos de la organización, y, como resultado, permitiendo incrementar el porcentaje de éxito de los proyectos de almacenes de datos.

1.2 Lista de Publicaciones incluidas en esta Tesis

Esta sección presenta un compendio de artíulos que han sido elegidos para formar parte de esta Tesis debido a su relevancia y contribución. Cada uno de estos artículos se describe brevemente en esta sección, incluyendo su relación con la Tesis Doctoral y el capítulo en el que se incluye.

Capítulo 3

Maté, A. *and Trujillo, J. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. Information Systems, (**IS**). 2012. Factor de Impacto: 1.595*

La trazabilidad es un tema relevante en almacenes de datos, dado que los almacenes de datos son estructuras complejas en constante evolución. Cada vez que un requisito o fuente es añadido o cambiado, tiene un cierto impacto en el almacén de datos. La mayoría de los cambios que se realizan ocasionan una pérdida de trazabilidad implícita entre los distintos modelos del almacén de datos, por lo que la capacidad del diseñador para validar el almacén de datos resultante se ve mermada. Además, como el almacén de datos es uno de los elementos núcleo en los sistemas actuales de Inteligencia de Negocio, cualquier retraso o error en el proceso afectará a la capacidad de toma de decisiones de la empresa.

En este capítulo inicial analizamos el soporte para trazabilidad que se provee en las propuestas actuales de almacenes de datos. Nuestro análisis muestra que las aproximaciones actuales asumen la existencia de trazabilidad como conocimiento implícito, en lugar de modelarlo de forma explícita. Sin embargo, tradicionalmente el modelado explícito de la trazabilidad ha ocasionado un sobrecoste, derivado de la creación y mantenimiento de trazas. Dado que el desarrollo de un almacén de datos es de por sí un proceso largo y costoso, cualquier sobrecoste introducido debe de ser minimizado. Para evitar este problema, en este trabajo describimos una propuesta de trazabilidad que puede integrarse en las aproximaciones para el desarrollo de almacenes de datos actuales y que considera el conjunto completo de modelos y relaciones que están involucrados en el proceso. Nuestra propuesta permite al diseñador rastrear de forma precisa cualquier requisito hasta el almacén de datos y viceversa, por lo que puede validar adecuadamente el cumplimiento de los requisitos y evaluar el impacto que puedan tener distintos cambios en el almacén. Además, tal y como se muestra, nuestra propuesta no introduce sobrecostes cuando se incluye dentro del marco de Desarrollo Dirigido por Modelos, y, además, permite automatizar distintas tareas tales como el análisis del impacto de cambios, la propagación de los mismos o la validación de requisitos.

Capítulo 4

Maté, A. *and Trujillo, J. Incorporating Traceability in Conceptual Models for Data Warehouses MDA. Proceedings of 30th International Conference on Conceptual Modeling (**ER'11**). 2011. Brussels, Belgium. Acceptance rate: 24.8%. ERA A*

Maté, A. *and Trujillo, J. and De Gregorio, E. and Song, I. Y. Improving the Maintainability of Data Warehouse Designs: Modeling Relationships between Sources and User Concepts. Proceedings of 15th Workshop on Data Warehousing and OLAP, (**DOLAP 2012**). 2012. Maui, Hawaii. Acceptance rate: 30%. ERA B*

El análisis realizado en el capítulo anterior muestra que el proceso de integración de datos y reconciliación del almacén es considerado de manera meticulosa por el diseñador mientras modela el almacén de datos siguiendo una aproximación supply-driven o híbrida. Sin embargo, los marcos de desarrollo de almacenes de datos actuales no proveen de herramientas para modelar esta integración. En lugar de ello, primero este proceso es analizado mientras el almacén de datos está siendo desarrollado y, entonces, se modela de forma separada al almacén mediante herramientas de Extracción/Transformación/Carga (ETL) una vez que el almacén ya ha sido implementado.

No obstante, los flujos de integración de datos representan una parte crucial del almacén de datos y de la estructura del sistema de Inteligencia de Negocio. Cualquier cambio en los flujos de datos inmediatamente pasa al almacén, afectando a varios requisitos, informes, y cuadros de mando. Por ejemplo, cambiar la columna de una fuente de datos puede llegar a afectar a más de 40 informes distintos, cuadros de mando y otras interfaces de los tomadores de decisión [20]. Así pues, es importante ser capaz de identificar de manera precisa el efecto de cualquier cambio en el almacén de datos y reacomodar la implementacion del almacén de datos. Por ello, en este capítulo comenzamos analizando como las relaciones entre requisitos y las estructuras de datos afectan al almacén de datos resultante y a sus requisitos originales. Identificamos dos tipos potenciales de relaciones *Overlap* y *Conflict* que afectan a cómo se deriva la información de contexto del almacén de datos (i.e. dimensiones). Por un lado, un overlap representa una estructura compatible en cuanto a la dimensión y a su jerarquía entre los requisitos y las fuentes de datos. Por otro lado, un conflicto representa una estructura incompatible, y resalta la necesita de considerar y decidir qué estrucutra debe adoptar finalmente la dimensión. De acuerdo a estas relaciones identificadas, definimos un conjunto de reglas Query/View/Transformation (QVT) [28] para derivar el esquema final del almacén de datos.

En segundo lugar, proponemos una formalización que permite modelar las relaciones identificadas durante el proceso de reconciliación entre el almacén de datos y las fuetnes de datos. Nuestra formalización extiende las relaciones previamente identificadas en un conjunto más detallado de trazas. Esta aproximación permite al diseñador modelar y documentar toda la información identificada durante el proceso de reconciliación, de forma que se provee información precisa acerca del mismo y resulta más sencillo (i) integrar nuevas fuentes de datos, (ii) analizar el grado de satisfacción de los requisitos de usuario, y (iii) especificar posteriormente procesos ETL. Como resultado, se mejora la mantenibilidad del almacén y se evita realizar tareas repetitivas y propensas a errores, tales como la inspección de fuentes y la fusión del diseño del almacén de datos con la información de las fuentes, que deberían ser repetidas cada vez que se realiza un cambio.

Capítulo 5

Maté, A. *and Trujillo, J. and Franch, X. Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star. Journal of Systems and Software (**JSS**). (In Press) Factor de Impacto: 1.135*

Los capítulos anteriores han tratado con los desafíos relacionados con la complejidad del desarrollo de almacenes de datos y la validación del esquema del almacén. Para ello, se ha introducido una infraestructura de trazabilidad a lo largo de los distintos modelos involucrados en el desarrollo. Sin embargo, aún es necesario tratar el problema de la validación de los requisitos del almacenes de datos. Para ello, trataremos este problema en dos partes. Primero nos focalizaremos en mejorar los modelos de requisitos utilizados, tema en el cual se centra el capítulo actual. Segundo, trataremos el problema de las vistas parciales de requisitos y la validación de los mismos en el capítulo siguiente.

La mayoría de aproximaciones recientes para el desarrollo de almacenes de datos hacen uso del modelado de objetivos para elicitar y modelar los requisitos de los usuarios. Entre los modelos de objetivos, uno de los más populares que se ha aplicado a distintas áreas, incluyendo los almacenes de datos, ha sido i* [42]. Los modelos de objetivos presentan la ventaja de que son más fácilmente entendibles por los tomadores de decisión. Sin embargo, i* presenta un problema de escalabilidad cuando los modelos se hacen demasiado grandes, ya que no proporciona ningún mecanismo de modularización [7]. En consecuencia, la validación de los requisitos se ve mermada, ya que los diagramas se vuelven más difíciles de utilizar para la comunicación entre diseñadores y usuarios.

En este capítulo analizamos la efectividad de las propuestas actuales basadas en i* para el modelado de requisitos en almacenes de datos. Primero tratamos el problema de la escalabidad, mediante la adición de un conjunto semántico de módulos al metamodelo [23, 8], permitiendo así al diseñador particionar mejor y presentar los modelos a los usuarios. A continuación llevamos a cabo un experimento para comparar la efectividad de los modelos con y sin módulos. Los resultados muestran una tendencia a disminuir el ratio de errores conforme los modelos se hacen más grandes, así como un aumento en el número de elementos identificados cuando se modelan los requisitos del almacén.

Capítulo 6

Maté, A. *and Trujillo, J. and Yu, E. Aligning Data Warehouse Requirements with Business Goals. Proceedings of the Sixth International i* Workshop (iStar 2013), CEUR Vol. 978, pp. 67-72. 2013 Valencia, Spain.*

Una vez que hemos mejorado los modelos de requisitos para facilitar la comunicación con los usuarios, nos centramos en el la problemática de la validación de los requisitos de usuario.

Las aproximaciones recientes para el desarrollo de almacenes de datos [25, 9] elicitan los requitios de los tomadores de decisión para dar soporte al proceso de toma de decisiones de la empresa. Sin embargo, estas aproximaciones (i) cubren únicamente los objetivos de los tomadores de decisión individuales, y (ii) sólo hacen uso del modelado de los objetivos del negocio para identificar qué información relevante debería incluirse en el almacén de datos. Por ello, los requisitos elicitados (i) sólo proveen vistas parciales del sistema, y (ii) no contribuyen a reducir la brecha entre expertos TI y expertos del negocio, ya que los diseñadores no pueden interpretar adecuadamente los requisitos en términos de negocio [4]. Por esta razón, en este capítulo vamos un paso más lejos que el estado del arte actual e (i) incorporamos la información del plan de negocio en la validación de requisitos del almacén de datos, y (ii) conectamos el almacén de datos con un modelo estratégico del negocio.

En este capítulo definimos un proceso para validar cada requisito de usuario contra la información guardada en el plan de negocio. Para conseguir este objetivo, primero representamos el plan de negocio utilizando el Business Intelligence Model (BIM) [3]. A continuación, definimos un conjunto de restricciones que deben cumplirse para que un requisito de negocio esté alineado con la estrategia. El resultado final es un almacén de datos alineado con la estrategia del negocio. Además, nuestra propuesta nos permite identificar los objetivos de negocio que se han pasado por alto en el sistema de Inteligencia de Negocio, así como los diferentes tomadores de decisión involucrados en la toma de decisiones con respecto al mismo objetivo de negocio, permitiendo comparar la información utilizada por cada uno y proporcionar una solución más integrada y completa.

Apéndice A

Maté, A. *and Trujillo, J. Tracing Conceptual Models Evolution in Data Warehouses by using MDA. Computer Standards and Interfaces. Under review. (2nd round) Factor de Impacto: 0.978*

En este capítulo se presenta una extensión de nuestro trabajo para documentar las relaciones entre el almacén de datos y las diferentes fuentes de datos. Este capítulo incluye una formalización de los tipos básicos de trazas que intervienen en el proceso de reconciliación, así como un conjunto mejorado de reglas QVT que permiten derivar el esquema final de almacén de datos a partir de cualquier configuración de relaciones entre requisitos y fuentes. De esta forma, se permite una rápida reconfiguración y análisis del almacén de datos cada vez que se cambia o se añade una fuente de datos.

1.3 Otras publicaciones en Conferencias internacionales

En esta sección se presenta un conjunto de artículos que han sido publicados como parte de la investigación realizada durante la Tesis Doctoral. Sin embargo, estos documentos no se han incluido en este compendio ya que son complementarios al núcleo de la tesis doctoral.

Maté, A. *and Trujillo, J. and Franch, X. A modularization proposal for goal-oriented analysis of data warehouses using i-star. Proceedings of 30th International Conference on Conceptual Modeling (**ER'11**). Lecture Notes in Computer Science Vol. 6998, pp. 421-428. 2011. Brussels, Belgium. Acceptance rate: 24.8%. ERA A*

*Franch, X. and **Maté, A.** and Trujillo, J. and Cares, C. On the joint use of i* with other modelling frameworks: A vision paper. Proceedings of the 19th IEEE International Requirements Engineering Conference (**RE**). 2011. Trento, Italy. Acceptance rate: 16,7%. ERA A*

Maté, A. *and Trujillo, J. and Mylopoulos, J. Conceptualizing and Specifying Key Performance Indicators in Business Strategy Models. Proceedings of 31th International Conference on Conceptual Modeling, (**ER'12**). Lecture Notes in Computer Science, Vol. 7532, pp. 282-291 2012. Florence, Italy. Acceptance rate: 26,2%. ERA A*

*Trujillo, J. and **Maté, A.** Business Intelligence 2.0: A General Overview. Lecture Notes in Business Information Processing Vol. 96, pp. 98-116. 2012. Springer.*

Maté, A. *and Llorens, H. and de Gregorio, E. An Integrated Multidimensional Modeling Approach to Access Big Data in Business Intelligence Platforms. Proceedings of the First International Workshop on Modeling for Data-Intensive Computing. Lecture Notes in Computer Science Vol. 7518, pp. 111-120. 2012 Florence, Italy.*

Maté, A. *and Trujillo, J. and Mylopoulos, J. Conceptualizing and Specifying Key Performance Indicators in Business Strategy Models. Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research (CASCON'12). pp. 102-115. 2012 Toronto, Canada*

Maté, A. *and de Gregorio, E. and Cámara, J. and Trujillo, J. Improving Massive Open Online Courses Analysis by applying Modeling and Text Mining: a Case Study.* Proceedings of the First International Workshop on Modeling and Management of Big Data. 2013 (In Press)

1.4 Sumario de la Tesis

El objetivo principal de esta Tesis Doctoral es definir una serie de técnicas para mejorar las aproximaciones existentes de desarrollo de almacenes de datos, cubriendo as los defectos de las propuestas anteriores. El objetivo de los almacenes de datos es dar soporte el proceso de toma de decisiones mediante la integración de varias fuentes de datos heterogéneas en una única verdad para la organización decisión. El desarrollo de almacenes de datos es un proceso largo y complejo que puede tardar años en completarse. A pesar de este esfuerzos, los proyectos de almacenes de datos rara vez satisfacen las necesidades de los usuarios, llegando a fracasar más del 70% de las veces, según estudios recientes [4].

Dada la especial idiosincrasia de los almacenes de datos se requiere una metodología de desarrollo específica que se diferencia de las metodologías de desarrollo de bases de datos y software tradicionales. Existen varias diferencias:

1. En primer lugar, un almacén de datos se modela conceptualmente en términos de hechos, centro del análisis, y dimensiones, contexto del análisis.

2. En segundo lugar, La estructura del almacén de datos, en términos de hechos y dimensiones, depende, por lo general, de la información que se los tomadores de decisión desean analizar. Sin embargo, la información que se almacena en el almacén de datos y su estructura dependen también de la disponibilidad de dicha información en las fuentes de datos, por lo que no todos los requisitos de los usuarios pueden ser satisfechos.

3. En tercer lugar, el proceso de toma de decisiones requiere datos recientes, por lo que el almacén de datos debe mantenerse al día con los cambios no sólo en las necesidades del usuario, sino también en las fuentes de datos de donde se extrae la información.

4. Por último, no toda la información se puede almacenarse en un solo esquema de OLAP (a menudo identificado con un cubo de análisis), ya que (i) los distintos usuarios están interesados en información diferente, y (ii) la complejidad de la información que se muestra aumenta conforme se añade más información, perjudicando así la comprensibilidad de los datos que se muestran y afectando al rendimiento de las consultas.

Para dar respuesta a estas características especifícas, en la literatura se han propuesto aproximaciones demand-driven, supply-driven e híbridas [34, 15, 9, 25, 18, 30, 5]. Una visión general de los pasos involucrados en estas aproximaciones puede verse en la Figura 1.1. En primer lugar, tanto en las aproximaciones demand-driven como híbridas, se recogen los requisitos de los usuarios. Entonces, de acuerdo a estos requisitos, se diseña un esquema multidimensional inicial. En el caso de las aproximaciones demand-driven puras, este esquema es implementado y cargado con datos. Por otro lado, las aproximaciones híbridas realizan una inspección detallada de las fuentes de datos y a continuación remodelan el esquema inicial de acuerdo a la información disponible. En el caso de las aproximaciones supply-driven puras, la inspección

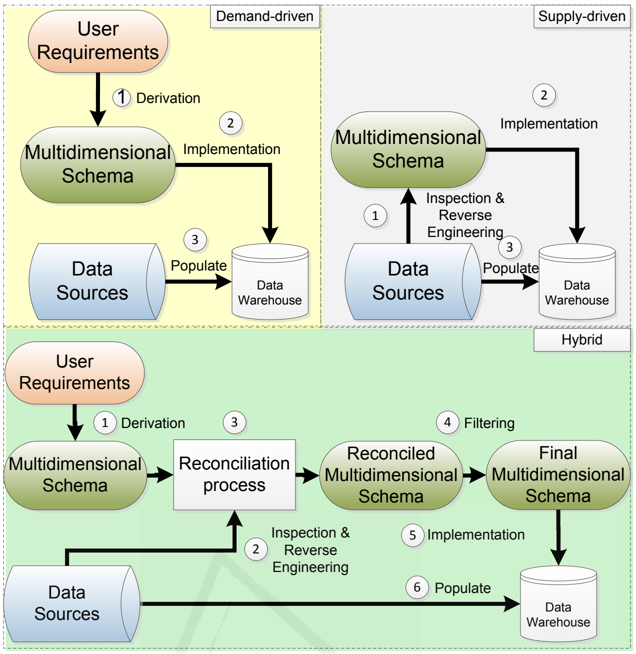


Figure 1.1: Overview of steps included in demand-driven, supply-driven and hybrid approaches.

de las fuentes de datos provee de los conceptos que se modelan en el esquema multidimensional del almacén. Finalmente, se implementa el almacén se datos de acuerdo con este esquema y se carga con los datos correspondientes.

Cada una de las técnicas presentadas tiene distintos inconvenientes. Los enfoques demand-driven requieren que el almacén de datos a diseñar sea relativamente pequeño y que haya un conocimiento detallado de la información almacenada en las fuentes de datos [9] con el fin de tener éxito. Por otra parte, las aproximaciones supply-driven ignoran mayoritariamente las necesidades del usuario, por lo que es posible que los tomadores de decisiones encuentren dificultades para entender el esquema que están analizando. Además, en ambos enfoques es posible que se pierda información importante, ya sea porque los tomadores de decisiones no pueden proporcionar una lista exhaustiva de toda la información que necesitan [41] o porque, al ignorar los requisitos de los usuarios, hay entidades y atributos que debería haber sido derivado de la información existente pero no se tuvieron en cuenta. Estos inconvenientes han llevado a la aparición de técnicas híbridas en los últimos años [25, 9]. Aunque los enfoques híbridos resuelven los problemas mencionados anteriormente, sufren de una pérdida de la trazabilidad derivada del proceso de reconciliación, como veremos en la sección siguiente.

1.4.1 Un Metamodelo de Trazabilidad para Trazar los Requisitos de los Usuarios

Para poder validar adecuadamente los requisitos de los usuarios y estimar la adecuación del almacén propuesto, es necesario ser capaz de trazar cada requisito hasta su implementación final. El primer capítulo de esta Tesis se centra en tratar el problema de preservar la trazabilidad entre los requisitos de los usuarios y el resto de modelos involucrados en el diseño del almacén de datos. Con esta meta, nuestro primer objetivo es elaborar un metamodelo de trazas adaptado para almacenes de datos que permita crear y preservar la información de trazabilidad.

Las aproximaciones recientes de almacenes de datos [25, 9] hacen uso de i* [42] para elicitar y modelar los requisitos de los usuarios. Esta aproximación tiene como objetivo disminuir la brecha entre expertos TI y tomadores de decisión, permitiendo al diseñador del almacén de datos extraer, mediante entrevistas, la información que los tomadores de decisión quieren incluir en el almacén de datos.

Sin embargo, en la práctica los diagramas de requisitos contienen un elevado número de elementos y relaciones, entre los que se incluyen los objetivos de los usuarios y la informacin a almacenar en el almacen entre otros. Todos estos elementos deben ser trazados a su correspondiente estructura multidimensional, lo cual puede resultar costoso si se realiza de forma manual.

Además, durante el proceso de reconciliación, el diseñador del almacén remodela las distintas estructuras del almacén de datos de diseño con el fin de permitir dar soporte a la mayor cantidad de requisitos y datos relevantes. Dado que los tomadores de decisión utilizan un lenguaje distinto a los de los diseñadores de las fuentes de datos, es habitual que los mismos conceptos utilicen nombres distintos y presentan una estructura distinta.

Por tanto, con el fin de mantener la trazabilidad en estos casos, o bien todos los modelos deben mantenerse sincronizado conforme se llevan a cabo cambios durante el proceso de reconciliación, o, al menos, los elementos deben estar explícitamente relacionados con el fin de ser permitir rastrear cada requisito hasta el conjunto correcto de elementos multidimensionales.

No obstante, las aproximaciones actuales de desarrollo de almacenes bien (i) no hacen referencia a estas relaciones [41, 5], bien (ii) se refieren a la existencia de estas relaciones entre modelos sin modelarlas explícitamente [9, 38], o bien (iii) proveen trazabilidad implícita mediante coincidencia de nombre [25]. Incluso en el mejor de los casos, la trazabilidad implícita sólo está garantizada hasta que se lleva a cabo la reconciliación de los requisitos con las fuentes de datos.

Por ello, al seguir las aproximaciones actuales para el desarrollo de almacenes de datos, no podemos garantizar que el diseñador sea capaz de verificar que la implementación del almacén de datos cumple con los requisitos de los usuarios. Para tratar este problema, nuestra propuesta es modelar explícitamente la trazabilidad desde los requisitos de los usuarios hasta las distintas estructuras multidimensionales que componen el almacén. De esta forma, evitamos los inconvenientes mencionados anteriormente y podemos evaluar automáticamente el estado de cada requisito.

Tradicionalmente, se puede diferenciar entre dos tipos distintos de trazabilidad. En primer lugar se encuentra la trazabilidad sobre un sólo modelo conforme evoluciona a lo largo del tiempo. Este tipo de trazabilidad registra las operaciones que se han realizado sobre el modelo en términos de adiciones, modificaciones y borrados. En segundo luga, tenemos la trazabilidad punto a punto, que rastrea las relaciones entre elementos en distintos modelos. Ejemplos de estas relaciones son las que existen entre los requisitos de usuario y las estructuras multidimensionales o las que se dan cuando se deriva un esquema relacional a partir de un esquema Entidad-Relación. Este tipo de trazabilidad es ideal para el desarrollo de almacenes de datos dado que estamos interesados en evaluar la adecuación del almacén de datos con respecto a los requisitos de usuario, por lo que tendremos que navegar a través de múltiples modelos.

Para dar soporte a la trazabilidad durante el proceso de desarrollo, primero identificamos los tipos de relaciones que existen entre los elementos en los distintos modelos del almacén. Utilizando la aproximación definida en [25] como un ejemplo de propuesta híbrida de desarrollo, podemos identificar 4 pasos modelo a modelo, mostrados en la Figura 1.2. En esta Figura, los requisitos de los usuarios son recogidos, modelados [24], y a continuación transformados en un modelo multidimensional inicial del almacén (1). Este modelo es a con-

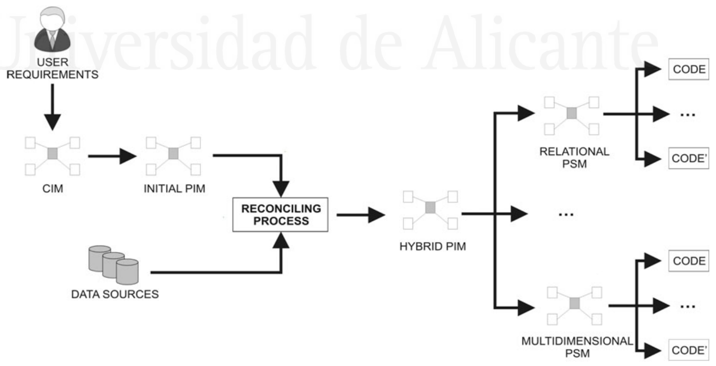


Figure 1.2: An example of hybrid development approach from [25].

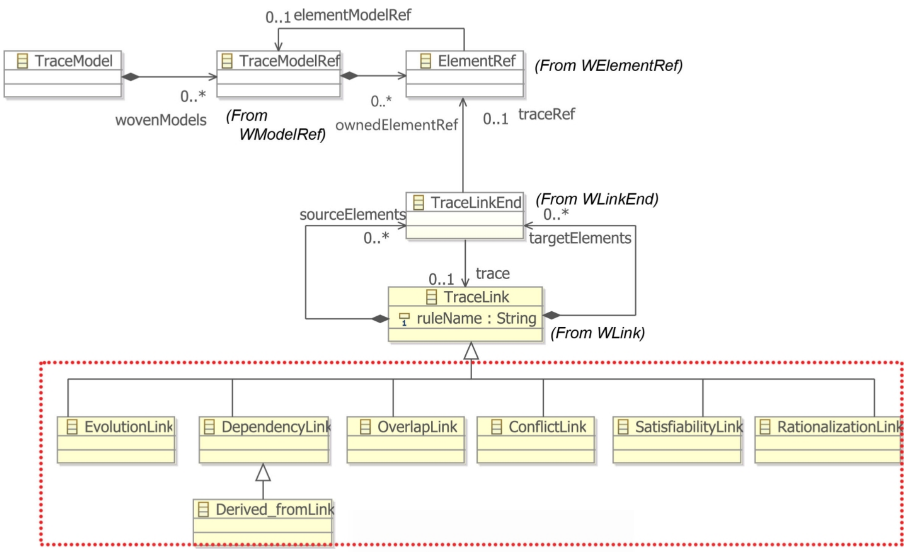


Figure 1.3: Traceability metamodel for explicit model of traceability in data warehouses [22]

tinuación reconciliado con las fuentes de datos, que están representadas en un modelo lógico compuesto de tablas (2). El resultado, es un modelo híbrido multidimensional que incluye información tanto de requisitos de usuario como de fuentes de datos (3). Finalmente, los elementos no deseados son filtrados o modificados y se obtiene el esquema final del almacén de datos (4).

Para categorizar las distintas trazas creadas durante este proceso, realizamos una revisión de la literatura acerca de trazabilidad tanto en Ingeniería de Requisitos (IR) como en Desarrollo Dirigido por Modelos (DDM) [36, 1, 2, 12, 11, 31, 43, 1, 16, 29, 39, 40]. El resultado de esta revisión fue una clasificación de las trazas de acuerdo a ocho categorías distintas utilizadas en IR [36]. A continuación, esta clasificación fue alineada con el proceso de desarrollo del almacén de datos en el Capítulo 3 y especializada en el Capítulo 4. La clasificación inicial se muestra en la Figura 1.3 y se compone de seis tipos semánticos básicos:

1. **Satisfiability** corresponde con la relación entre las requisitos de usuario y los elementos multidimensionales. Su origen debe ser siempre un requisito que tiene uno o más elementos multidimensionales que lo satisfacen. Estos homólogos multidimensionales son siempre los objetivos del enlace, ya que su existencia está ligada a los requisitos del usuario. En nuestra aproximación, el único elemento del diagrama que no tiene una

contrapartida multidimensional son los objetivos del usuario. Como su satisfacción depende de la existencia de la información necesaria para la toma de decisiones, pueden ser evaluados comprobando si los elementos trazados en el almacén de datos se encuentran implementados.

2. **Derived_from** captura la relación entre fuentes de datos y los elementos multidimensionales. Su fuente debe ser siempre uno o más elementos de las fuentes de datos, tales como tablas, columnas, claves, etc. representados a nivel lógico, y sus elementos objetivo son uno o más elementos multidimensionales obtenidos mediante ingeniería inversa del modelo lógico.

3. **Evolution** corresponde con la relación entre los elementos de un modelo multidimensional y su versión correspondiente en el modelo siguiente. Estos enlaces están diseñados especialmente para enfoques tales como [25], en donde hay múltiples modelos multidimensionales involucrados en el proceso de desarrollo. Por ejemplo, la existencia (o ausencia) de un enlace de evolución desde uno de los elementos del modelo de reconcilición hasta el modelo final determina si ese elemnto fue efectivamente seleccionado para ser incluido en la implementación almacén de datos o se descartó.

4. **Overlap** representa coincidencias entre los elementos multidimensionales esperados por los requisitos de usuario y aquellos obtenidos mediante ingeniería inversa a partir de las fuentes de datos. Este tipo de relaciones se analizará con más detalle en la sección siguiente.

5. **Conflict** representa diferencias entre los elementos multidimensionales esperados por los requisitos de usuario y aquellos obtenidos mediante ingeniería inversa a partir de las fuentes de datos. Este tipo de relaciones se analizará con más detalle en la sección siguiente.

6. **Rationalization** captura las relaciones que no están cubiertas en un desarrollo modelo de almacenes de datos. Por ejemplo, permiten relacionar elementos solución de conflictos con los elementos originales, así como trazar otros añadidos realizados por el diseñador.

Una vez que hemos definido las categorías de trazas, tenemos que definir un metamodelo de trazabilidad o adaptar uno existente que sea (i) débilmente acoplado con los modelos del almacén de datos, ya que no existe ningún estándar para el modelado de requisitos de los usuarios ni para el modelado multidimensional hasta ahora, por lo que de otra forma limitaría la aplicabilidad de nuestra aproximación, y (ii) evite contaminar los modelos con la información de trazabilidad, ya que puede ser confusa para el diseñador de almacén de datos [16]. Después de realizar una revisión de las propuestas existentes, obtuvimos el metamodelo de trazabilidad presentado en la Figura 1.3, que especializa la propuesta general de [6].

El metamodelo de trazabilidad mostrado es altamente flexible, ya que permite la definición de trazas muchos a muchos involucrando tantos modelos como sea necesario. Además, el modelo es débilmente acoplado con los modelos trazados, ya que funciona mediante el uso de referencias a elemento y a modelo, evitando así la inserción de información de trazabilidad en los modelos del almacén de datos. Por ello, puede ser aplicado para mejorar cualquiera de las propuestas de desarrollo de almacenes de datos existentes.

Sin embargo, la trazabilidad puede introducir una sobrecarga considerable en cualquier proceso si tiene que ser capturada y mantenida de forma manual. Por tanto, con el fin de evitar la introducción de sobrecarga en un proceso de desarrollo ya de por sí costoso, integramos la generación de trazas en el marco de una aproximación basada en DDM. De esta forma, generamos las trazas automáticamente de forma simultánea a la derivación del modelo del almacén de datos en el siguiente paso del proceso, tal y como se muestra en más detalle en el Capítulo 3. Mediante la introducción de la trazabilidad en el proceso base, obtenemos un proceso de desarrollo refinado que incluye un conjunto de modelos de trazas por encima de los modelos tradicionales para el desarrollo de almacenes de datos. El conjunto completo de modelos se presenta en la Figura 1.4. En esta Figura, podemos ver el almacén de datos y los distintos modelos de trazabilidad desde la perspectiva de DDM. Los requisitos se recogen en la capa de CIM (Computation Independent Model). A continuación, se derivan

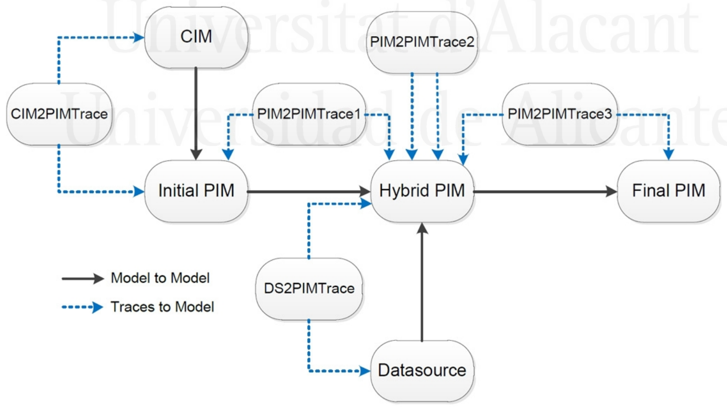


Figure 1.4: End-to-End traceability models included into the hybrid development process.

en los modelos multidimensionales en la capa de PIM (Platform Independent Model). Finalmente, estos modelos se reconcilian con la información de las fuentes de datos, representada a nivel PSM (Platform Specific Model) y se obtiene el modelo multidimensional final.

Los nuevos modelos de trazabilidad introducidos enlazan los requisitos de usuario y las fuentes de datos con los modelos multidimensionales del almacén. Esto nos permite trazar elementos de los diagramas multidimensionalesde y de las fuentes de datos hasta los requisitos de los usuarios, e identificar (i) qué requisitos de usuario no pueden ser satisfechos, y (ii) qué elementos se verán afectados por un cambio y propagar el cambio si el diseñador lo considera oportuno. Estos modelos se generan de forma automática mediante reglas Query/View/Transformation (QVT) [28]. QVT es un estándar propuesto por el Object Management Group para la creación de transformaciones modelo a modelo. Una regla QVT, tal como la mostrada en la Figura 1.5 comprueba la existencia del patrón localizado el lado izquierdo de la regla. Si se encuentra dicho patrón, entonces se ejecuta la parte derecha de la regla, que lleva a cabo la creación de los elementos correspondientes en cada modelo. La implementación de estas reglas QVT en nuestra herramienta CASE se realiza mediante el lenguaje ATLAS Transformation Language (ATL) [17].

Conceptualmente, los requisitos de los usuarios están enlazados a los elementos multidimensionales, tal y como se muestra en la Figura 1.6. Estos

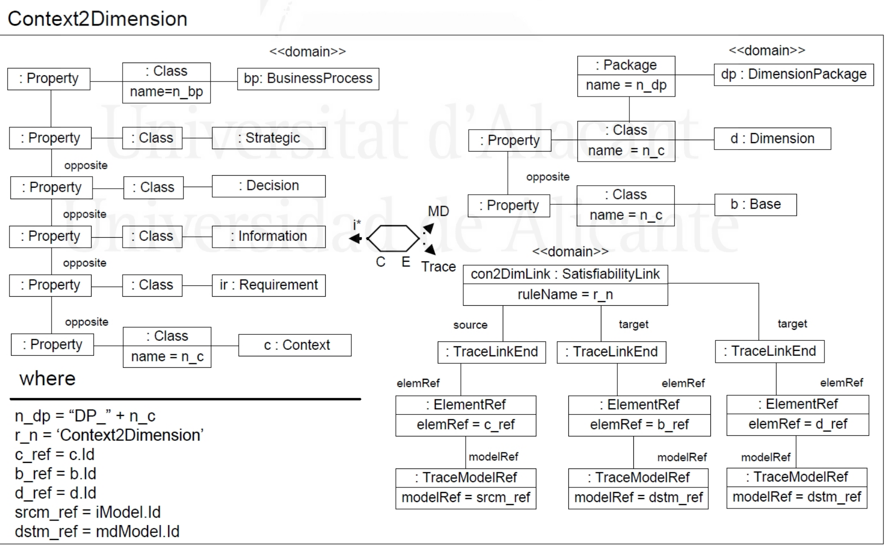


Figure 1.5: A QVT rule enhanced with traceability information.

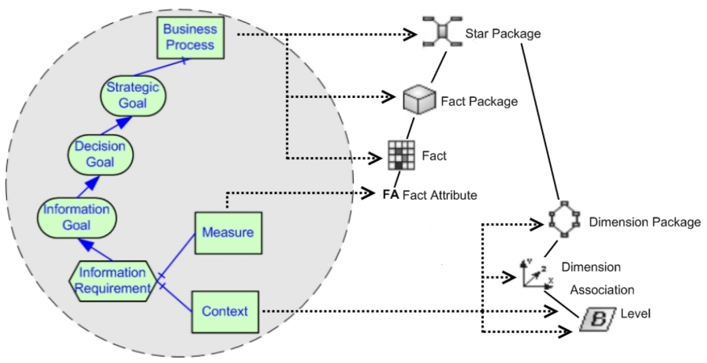


Figure 1.6: Conceptual trace links between user requirements and multidimensional models.

enlaces se mantienen actualizados a un framework reactivo que observa y registra los cambios realizados en los modelos y permite propagarlos a los distintos modelos, incluyendo la actualización automática de modelos de trazas. En la práctica, los modelos de trazas pueden ser inspeccionados por el diseñador del almacén de datos, así como ser utilizados como entrada de otras transformaciones modelo a modelo que requieran de información de trazabilidad. Sin embargo, para el diseño de almacenes de datos, su objetivo es ser transparentes para el usuario, y dar soporte para la realización de diferentes tareas, tales como la propagación de cambios o el análisis del impacto de cambios, como se muestra en el Capítulo 3.

1.4.2 Trazabilidad durante el Proceso de Reconciliación

A fin de trazar los requisitos de los usuarios hasta la implementación del almacén de datos, los requisitos tienen que ser trazados a través del proceso de reconciliación. Durante este paso, algunas estructuras multidimensionales pueden ser cambiadas o incluso descartadas por el diseñador, debido a las diferencias entre la estructura esperada de los datos y la estructura real. Por tanto, el paso de la reconciliación requiere una atención especial, ya que a menudo se deja a la experiencia del diseñador de almacenamiento de datos, y no se realiza de manera suficientemente sistemática ni detallada como para poder analizar y validar el razonamiento detrás del proceso.

El paso de la reconciliación toma como entrada un modelo multidimensional acorde a los requisitos de los usuarios y un conjunto de fuentes de datos para suministrar datos a ese modelo multidimensional. Los objetivos de este paso son: (i) verificar que los datos necesarios para poblar las estructuras del modelo multidimensional se encuentran disponibles, y (ii) encontrar datos adicionales que puedan ser relevantes para la toma de decisiones, pero que hayan sido pasados por alto durante la etapa de elicitación requisitos. Durante esta etapa, los elementos pueden ser o bien descartados, en caso de que no haya disponibles, o bien confirmados, o bien añadidos, ya que pueden ser relevantes para el proceso de toma de decisiones, o bien modificados porque los datos datos no puedan ser transformados acorde a lo esperado por los requisitos de los usuarios. Las aproximaciones actuales para el desarrollo del almacén de datos comprueban la disponibilidad de la información mediante coincidencia de nombres [25], mediante la aplicación de formas normales multidimensionales [26], al modelar la actividad del negocio [9], o haciendo uso de ontologías [32]. Sin embargo, cualquier desajuste entre el esquema del almacén de datos y las fuentes de datos que no siga estas pautas debe ser resuelto por el diseñador del almacén de datos.

Las técnicas actuales de almacenes de datos no consideran de manera explícita estos desajustes y, por tanto, no proveen de herramientas para modelarlos. Por ello, el diseñador tiene que realizar todas las operaciones de acuerdo a su propia experiencia, el esquema multidimensional esperado, y las fuentes de datos disponibles. Desafortunadamente, estos desajustes ocurren de manera habitual debido a varias razones:

1. Primero, la nomenclatura utilizada por los tomadores de decisión rara vez coincide con la utilizada en las fuentes de datos operacionales.

2. Segundo, es poco habitual que la estructura de los conceptos y atributos en el esquema del almacén de datos coincida con aquellos extraídos de las fuentes de datos, incluso cuando se lleva a cabo un proceso de ingeniería inversa para obtener una vista multidimensional de las mismas. Por ejemplo, considerese una base de datos con una tabla para almacenar los detalles de los pedidos y con una clave ajena a otra tabla que contenga la lista de pedidos en sí. Esta estructura es típica de muchas bases de datos transaccionales, y se puede ver, por ejemplo, en la base de datos de ejemplo Northwind Traders (Figura 1.7). La aplicación de formas normales multidimensionales a este tipo de fuentes da como resultado un esquema multidimensional incorrecto, identificando los pedidos como una dimensión y el cliente y el proveedor como niveles de jerarquía de dicha dimensión.

3. Tercero, no toda la información se encuentra hoy día de forma exclusiva en las fuentes de datos operacionales, por lo que el modelado de los procesos de negocio tiene un alcance limitado. Por ejemplo, uno de los desafíos actuales de las empresas es romper los silos de información, presentando cada uno de ellos distinta estructura para el almacenamiento de los datos.

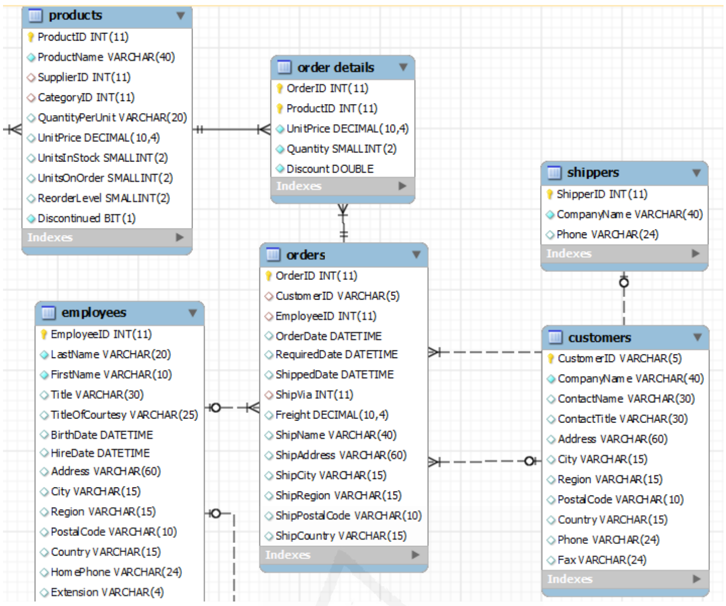


Figure 1.7: Excerpt of the transactional database Northwind Traders.

Como resultado de todos estos factores y de la falta de herramientas para documentar el proceso, el paso de reconciliación se comporta como una caja negra. Por lo tanto, presenta diversos inconvenientes, tales como la pérdida de trazabilidad, dificultades para intepretar la información proveída por las fuentes de datos y dificultades para integrar nuevas fuentes de datos, entre otras.

Con el fin de evitar estos inconvenientes, se propone un proceso para preservar la trazabilidad durante la reconciliación. Nuestra propuesta es mantener los elementos requeridos y los obtenidos por ingeniería inversa de forma separada. Entonces, en lugar de que el diseñador los fusione mediante combinaciones de operaciones de añadir, modificar y borrar, modelamos el de flujo de punto a punto de la información, que será implementado más tarde en los procesos de Extracción/Transformación/Carga (ETL) proceses. Las principales diferencias entre nuestro enfoque y los procesos ETL son que los enlaces son (i) establecidos a nivel conceptual con una visión multidimensional, (ii) creados en tiempo de diseño, mientras que los procesos ETL se crean después de realizar la implementación del almacén de datos, y (iii) que establecen conexiones punto a punto, pero no especifican cómo la información se transforma o se filtra. De esta forma, el diseñador puede modelar las relaciones entre los elementos requeridos y los obtenidos mediante ingeniería, y podemos ofrecer un análisis adicional acerca del tipo de desajustes presentes en la estructura del almacén de datos. Con esta finalidad, en primer lugar nos centraremos en la definición de las relaciones básicas entre elementos que afectan a la derivación del modelo reconciliado del almacén. A continuación, en la sección siguiente, analizaremos estas relaciones en profundidad. Las relaciones básicas posibles se definen como sigue:

1. Uno o más elementos de las fuentes de datos pueden proveer información a uno o más elementos del almacén sin requerir modificaciones. En este caso, las relación establecida entre los elementos se define como un *Overlap*. Como resultado, los requisitos de los usuarios satisfechos por los elementos multidimensionales están completamente soportados por la implementación del almacén.

2. Uno o más elementos de las fuentes de datos pueden proveer información a uno o más elementos del almacén necesitando ser modificados. En este caso, las relación establecida entre los elementos se define como un *Conflict*. Un ejemplo de estas modificaciones es cuando los datos se encuentran más agregados en el almacén de datos con respecto a las fuentes. Otro ejemplo se da cuando los datos están más agregados que lo requerido por el almacén de datos, por lo que la resolución del esquema del almacén ha de ser reducida para proveer, al menos, cierto grado de información.

3. Uno o más conflictos pueden ser solucionados creando un elemento reconciliado. En este caso, las relaciones establecidas entre los elementos se definen como una *Rationalization*, llevada a cabo por el diseñador.

Tras definir las relaciones básicas podemos relacionar los flujos de información entre el esquema esperado y el mediante ingeniería inversa. Por tanto, el siguiente paso en el proceso consiste en analizar cada elemento requerido y relacionarlo con los elementos correspondientes (si los hubiese) de la fuentes de datos que proporcionan la información necesaria. El resultado de este paso se puede verse ejemplificado en la Figura 1.8. En esta Figura, el esquema del almacén de datos esperado de una universidad se relaciona con el obtenido mediante ingeniería. Como podemos ver, ambas dimensiones están marcadas como en conflicto, debido a un desajuste en la forma en la que se identifican las instancias de sus niveles de jerarquía. Por otra parte, también podemos ver que el atributo "Name" del profesor se divide en tres atributos diferentes en las fuentes de datos y, por lo tanto, requiere una transformación. Aunque los nombres y la estructura de los dos esquemas no coinciden plenamente, obtenemos una visión clara de los elementos que intervienen en el proceso de reconciliación. En la siguiente sección vamos a proporcionar más detalles sobre la lógica de las relaciones y formalizaremos su significado.

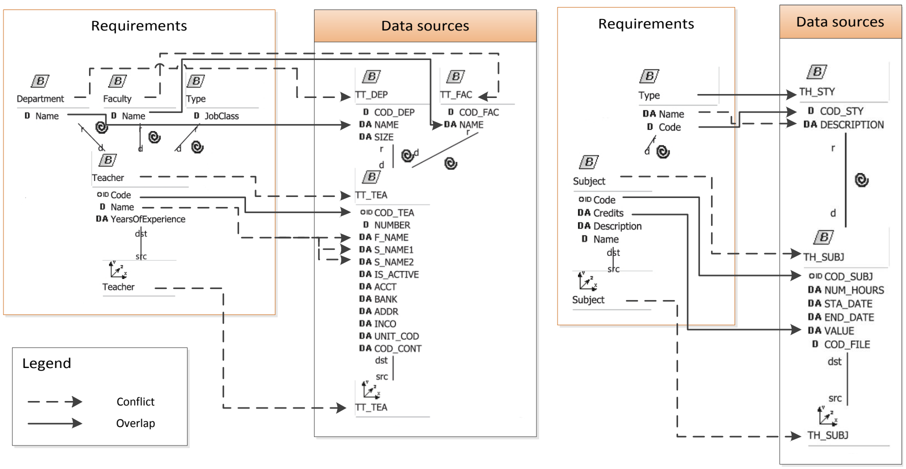


Figure 1.8: Example of mappings between the expected concepts and the reverse engineered ones

Una vez que todos los elementos se han relacionado, se revisan los conflictos existentes y ya o bien (i) se elige un elemento como solución, o (ii) se crea un elemento reconciliado y se relaciona con los elementos en conflicto por medio de un enlace *Rationalization*. Por último, seleccionamos los elementos que desea incluir en la implementación del almacén de datos. Con el fin de evitar la introducción de sobrecarga en el proceso, la creación del esquema multidimensional final puede automatizarse mediante el filtrado de los elementos no seleccionados y siguiendo dos reglas en el proceso de derivación:

1. Si dos o más elementos se encuentran relacionados mediante *Overlap*, si cualquiera de ellos se selecciona como candidato para el esquema final del almacén de datos, entonces se crea un elemento correspondiente y se trazan tanto los elementos requeridos como los obtenidos por ingeniería inversa hasta el esquema final por medio de trazas de *Evolution*.

2. Si dos o más elementos se encuentran relacionados mediante *Conflict*, si cualquiera de ellos se selecciona como candidato para el esquema final del almacén de datos, entonces se crea un elemento correspondiente y se trazan hasta el esquema final únicamente los elementos seleccionados por medio de trazas de *Evolution*.

Estas dos reglas pueden ser automatizadas tal y como se muestra en el Capítulo 4, mediante la codificación de su lógica en reglas QVT [28], automatizando completamente por tanto el proceso de derivación. Por otra parte, al seguir estas dos reglas, podemos identificar si un requisito de usuario se encuentra completamente soportado, soportado parcialmente o descartado en el diseño final de almacenamiento de datos. Los requisitos completamente soportados son aquellos que se pueden rastrear hasta elementos solapados o en conflicto que fueron seleccionados y derivados en el esquema final del almacén. En el primer caso, los elementos correspondiente de las fuentes de datos era compatible y por lo tanto los datos están disponibles. En el segundo caso, aunque el elemento de la fuente de datos correspondiente no es compatible, al seleccionar el elemento de los requisitos significa que se respeta su estructura, y, por lo tanto, se transformarán los datos acorde a ello. Además, si el diseñador proporciona un elemento reconciliados para un conflicto, también consideramos que el requisito de usuario está completamente soportado, ya que se proporcionó una solución al conflicto. Por otra parte, los requisitos parcialmente soportados son aquellos que se deben a elementos en conflicto cuyos homólogos en conflicto fueron seleccionados. Por lo tanto, los datos disponibles actúan como un sustituto de los datos esperados. Por último, los requisitos no soportados o descartados son los que no se puede rastrear en el esquema final de almacenamiento de datos a través de cualquier elemento.

Después de definir el proceso para preservar la trazabilidad a través de la reconciliación y derivar el esquema final de almacén de datos, procederemos a analizar en profundidad las relaciones entre las estructuras multidimensionales requeridas y aquellas obtenidas mediante ingeniería inversa y formalizarlas. Esto nos servirá para varios propósitos: (i) para obtener una mejor comprensión de la situación de los requisitos del usuario en el proceso de desarrollo, (ii) ser capaces de semi-automatizar la creación de trazas durante la reconciliation, y (iii) para proporcionar la diseñador del almacén de datos una semántica rica para documentar adecuadamente el proceso de reconciliación y servir como base para futuros procesos ETL.

Modelando la Integración de Datos durante la Reconciliación

Las relaciones definidas en la sección anterior están orientadas a preservar la trazabilidad. Sin embargo, no son lo suficientemente detalladas como para proporcionar las herramientas de diseño necesarias para documentar el paso de reconciliación. Los métodos actuales de desarrollo de datos de almacén se basan en los procesos de Extracción/Transformación/Carga (ETL) como medio para documentar los resultados de dicha etapa. Sin embargo, este enfoque tiene varios inconvenientes. En primer lugar, los procesos de ETL conectan fuentes de datos con el resultado de la etapa de reconciliación, una vez que el almacén de datos ya se ha implementado, en lugar de con el esquema del almacén de datos especificado por los requisitos de los usuarios. Por lo tanto, no pueden ser usados para evaluar si una nueva fuente de datos proporciona información previamente no disponible que permite satisfacer requisitos de usuario previamente insatisfechos. Además, esto también significa que no pueden documentar los desajuste ocurridos entre la información esperada y la que finalmente se almacenó en el almacén de datos. En segundo lugar, los procesos ETL especifican el flujo de información desde las fuentes de datos hasta las tablas en la base de datos de destino. Por tanto, ignoran las características multidimensionales de datos. En tercer lugar, los procesos ETL no proporcionan una visión global de las relaciones. Por el contrario, se dividen en varios archivos con diferentes intervalos de refresco. Por ello, la etapa de reconciliación queda pobremente documentada.

La falta de documentación puede convertirse en un problema cuando se lleva a cabo la incorporación de nuevas fuentes de datos en un almacén de datos existente. Si la documentación resulta insuficiente, puede ser necesaria una inspección manual de las fuentes de datos ya existentes, una tarea con un alto coste temporal y propensa a errores. Por desgracia, esta tarea es cada vez más común hoy en día. Por un lado, la tendencia de Big Data está ganando fuerza, lo que implica la adición de información de nuevas fuentes externas, tales como redes sociales o recursos RDF. Por otro lado, muchas empresas ya tienen un almacén de datos. Cuando una empresa se fusiona con otras, le resulta necesario integrar toda la información disponible en un almacén de datos único, por lo que se requiere información precisa sobre los datos almacenados. En la práctica, lo que sucede es que las empresas no son capaces de realizar esta tarea, y los silos de información comienzan a aparecer, dificultando el compartir información y comparar los datos a lo largo de la empresa.

Para hacer frente a este problema, especializamos las relaciones definidas para preservar la trazabilidad, permitiendo al diseñador del almacén de datos especificar la semántica concretas de las relaciones entre el almacén y las fuentes de datos. Para cubrir todas las posibilidades y proporcionar un conjunto completo de categorías, modelamos las relaciones desde el punto de vista de la Teoría de Conjuntos. En nuestra aproximación, nos referimos a los elementos multidimensionales especificados por los requisitos como elementos esperados o conceptos del usuario. Para cada concepto del usuario, el diseñador especifica un dominio D del cual el concepto puede tomar valores. A continuación, este conjunto de valores se compara con el dominio definido por los valores almacenados en las fuentes de datos. Como resultado de esta comparación se escoge una sola categoría de nuestra clasificación. La clasificación completa se muestra en la Figura 1.9, que especializa las categorías base *Overlap* y *Conflict*. Como línea de base, se define la relación *Overlap* como una coincidencia entre los dominios de origen y destino. Por el contrario, la relación *Conflict* implica un desajuste entre los dominios de origen y destino. Además, en los párrafos siguientes nos referiremos a los elementos en el almacén de datos como fuentes de las trazas y a los elementos de las fuentes de datos

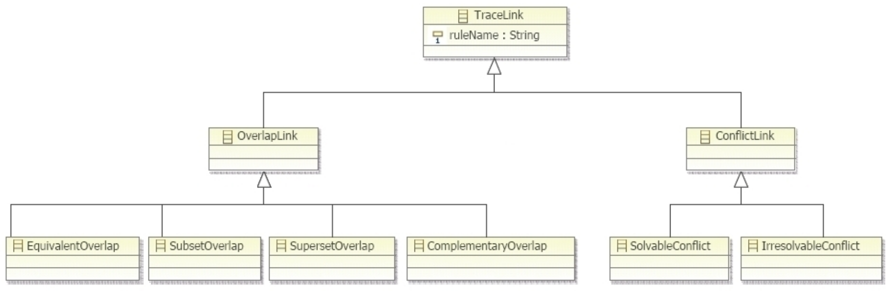


Figure 1.9: Classification of relationships between user concepts and data sources

como objetivos de las mismas[1].

1. Un *Equal Overlap* ocurre cuando tanto el dominio origen como objetivo comparten los mismos elementos.

2. Un *Subset Overlap* occurre cuando el dominio objetivo carece de ciertos elementos incluidos en el dominio origen. Por ejemplo, si esperamos que el identificador de un documento “idDocument” contenga identificadores de varias bibliotecas digitales pero los datos sólo contienen identificadores de una biblioteca.

3. Un *Superset Overlap* es la relación inversa de un *Subset Overlap*.

4. Un *Complementary Overlap* ocurre cuando tanto el dominio origen como el objetivo incluyen ciertos elementos que no aparecen en otros dominios.

5. Un *Solvable Conflict* occurre cuando el dominio origen y el dominio objetivo no coinciden, pero existe una función F que permite proyectar elementos desde el dominio objetivo al dominio origen. Por ejemplo, un atributo “language_code” no comparte el mismo dominio que el atributo “language”, pero puede ser transformado con la ayuda de una tabla de códigos de idiomas.

6. Un *Irresolvable Conflict* occurre cuando no existe una función F o es desconocida.

Cada una de estas categorías se puede aplicar a varios niveles de abstracción en los esquemas multidimensionales, desde los atributos hasta las dimensiones.

[1]En la práctica, es más fácil de comprobar sistemáticamente cada elemento en el esquema esperado del almacén de datos que comprobar cada elemento de las fuentes de datos

La formalización específica de cada categoría y nivel de abstracción se puede encontrar en el capítulo 4. Mediante el uso de este conjunto de categorías, el diseñador puede elegir especificar la naturaleza exacta de la relación entre el almacén de datos y las fuentes de datos durante la etapa de reconciliación. Por otra parte, sólo es necesario que el diseñador relacione el nivel de atributo, ya que el resto de los niveles de detalle se calculan automáticamente.

Hemos aplicado con éxito nuestra aproximación al caso de estudio de una biblioteca digital en la Universidad de Alicante. En nuestro caso, se deseaba construir un modelo de análisis de los documentos almacenados en la biblioteca digital. Sin embargo, debido a la evolución de las normas y ontologías utilizadas en las bibliotecas digitales, había información dispersa a través de varios atributos, niveles de jerarquía y dimensiones. Gracias a nuestro enfoque, pudimos obtener una visión clara de la información que era necesario recoger para cada dimensión requerida, si era necesario realizar una transformación o no, y qué información no se encontraba.

Por ejemplo, como se muestra en la Figura 1.10, necesitábamos información de dos dimensiones diferentes para obtener todos los datos necesarios para la dimensión "Document". Una de estas dimensiones contenía un nivel comple-

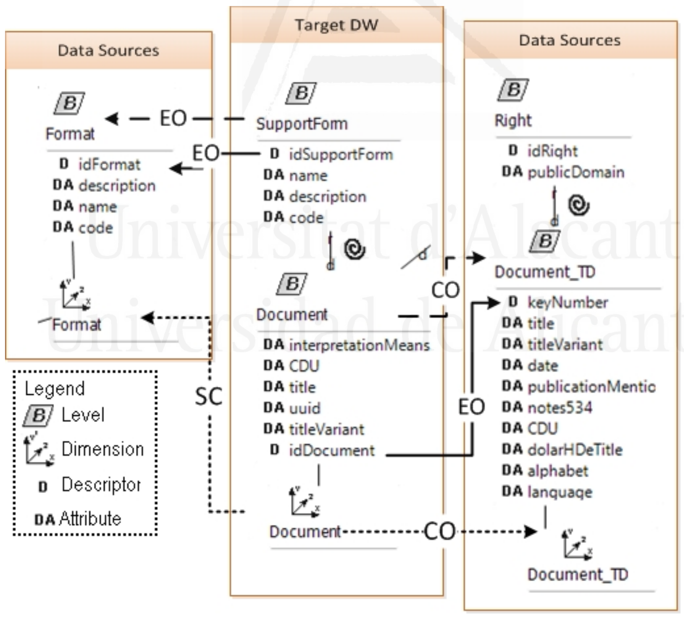


Figure 1.10: Example of detailed traceability between the expected schema and the reverse engineered from data sources.

mentario al primero que necesitábamos, ya que había algunos atributos que no estaban incluidos y, en su lugar, había otros atributos inesperados. Por otro lado, la otra dimensión contenía un nivel equivalente al segundo necesario. Para una descripción completa del estudio de caso, por favor consulte el Capítulo 4.

En comparación con los enfoques actuales, nuestra propuesta tiene varias ventajas. En primer lugar, todas las relaciones se agrupan en un único modelo, que proporciona una visión general de todos los elementos que intervienen en el proceso. Esto hace que sea más fácil planificar la integración de nuevas fuentes de datos en el proceso. En segundo lugar, podemos analizar con más precisión qué impacto tendrá sobre el almacén un cambio en las fuentes de datos. Por ejemplo, la modificación de los valores cargados en un descriptor (identificador) de nivel puede afectar en gran medida al almacén de datos, ya que altera los resultados obtenidos por la agregación. En tercer lugar, nuestra aproximación permite al diseñador resaltar información importante, por ejemplo, si cualquier elemento carece de cierta información. Por ejemplo, en algunos casos, los tomadores de decisión han informado de anomalías durante el análisis de los valores de los indicadores clave de rendimiento (KPI). Muchas veces, estas anomalías son debido a una falta parcial de datos en las fuentes de datos operacionales y, por lo tanto, el valor real que los tomadores de decisión esperan no se encuentra cargado en el almacén de datos.

Gracias a la combinación de nuestra formalización para la reconciliación y el metamodelo de trazabilidad, somos capaces de rastrear con precisión y evaluar el impacto de un cambio en cualquier requisito de los usuarios. Esto incluye no sólo aquellos cambios producidos en el almacén de datos, sino también aquellos realizados sobre las fuentes de datos. Por lo tanto, nuestra metodología mejorada para el desarrollo del almacén de datos puede soportar mejor la evolución constante de los almacenes y depender en menor medida en la experiencia del diseñador del almacén.

1.4.3 Analizando y Mejorando Diagramas de Requisitos

En las secciones anteriores hemos abordado en detalle el problema de la pérdida de la trazabilidad durante el desarrollo del almacén de datos. Así, hemos cubierto los objetivos iniciales de (2) Reducor la complejidad de la gestión de varios modelos involucrados en el desarrollo del almacén de datos, (3) Asegurar que los modelos de almacenamiento de datos resultantes cumplen adecuadamente los requisitos especificados, y (4) Dar soporte a la evolución y mantenimiento del almacén una vez que se ha implementado. Estos puntos han sido mejorados a las capacidades de navegación proporcionadas por las trazas, por el framework reactivo para la propagación de actualizaciones, y para el proceso de reconciliación modificado que permite al diseñador tener una mejor visión de la información involucrados en el proceso. Ahora, en las dos secciones siguientes nos centraremos en (1) Mejorar la validación requisitos.

A pesar de la inclusión de los modelos de requisitos basados en objetivos por parte de las aproximaciones más recientes como [25] y [9], la etapa de elicitación de requisitos aún presenta problemas. En primer lugar, los tomadores de decisión no son expertos técnicos, y mientras que los modelos de objetivos les resultan comprensibles, están más acostumbrados a pensar en términos de medidas e indicadores clave de rendimiento, en lugar de en términos de objetivos. Por lo tanto, es importante mantener los modelos simples o, al menos, focalizados, con el fin de elicitar adecuadamente los requisitos. Desafortunadamente, los enfoques recientes hacen uso de i* como el framework básico para la recopilación de requisitos, que carece de cualquier tipo de mecanismo de modularidad aparte de los elementos Actor/Rol. En consecuencia, el tamaño de los modelos se convierte en un problema para la corrección y la comunicación con los usuarios. En la práctica, el tamaño del diagrama aumenta hasta el punto de que incluso el diseñador del almacén de datos pasa por alto algunos de los elementos ya definidos y los duplica con diferente estructura. Por tanto, es importante mejorar este aspecto con el fin de poder gestionar correcciones y cambios en los requisitos del almacén de una forma más sencilla.

Proponemos abordar este problema por medio de la mejora del aspecto de modularidad de i* en el campo de almacenes de datos, permitiendo al diseñador particionar el diagrama siempre que sea necesario, mientras que al mismo tiempo mantenemos la semántica de cada partición. Para ello, extendemos el modelo de requisitos de i* para almacenes de datos [24] y definimos un conjunto de módulos con este objetivo. Primero presentamos los elementos que intervienen en la elicitación de requisitos por medio de un ejemplo, que se muestra en la Figura 1.11. A continuación, describimos el proceso básico que se lleva a cabo en la elicitación de requisitos y presentar nuestra extensión de módulos.

En nuestro ejemplo, comenzamos la elicitación de los requisitos a partir de un proceso de negocio (BP), relacionado con el tomador de decisiones. El BP, que es el centro del análisis, modela una actividad de interés para el tomador de decisiones. En este caso, la actividad es *Make Contracts*, y se asocia con una serie de objetivos estratégicos, que tienen como meta mejorar el rendimiento del negocio. Los objetivos estratégicos representan el nivel más alto de abstracción. Se consideran como cambios de una situación actual a otra mejor, en términos de los objetivos del proceso de negocio. En nuestro caso, los objetivos estratégicos relacionados con el BP son *Cost of contracts minimized* y *Quality of workers increased.* Otros ejemplos de objetivos estratégicos serían *Incrase sales*, *Increase number of customers*, *Decrease cost*, etc. Su cumplimiento produce un beneficio inmediato para la organización.

Para alcanzar estos objetivos estratégicos, hay una serie de objetivos decisionales que se deben cumplir. Los objetivos decisionales representan el nivel medio de abstracción en nuestros modelos SR (Strategic Rationale). Estos

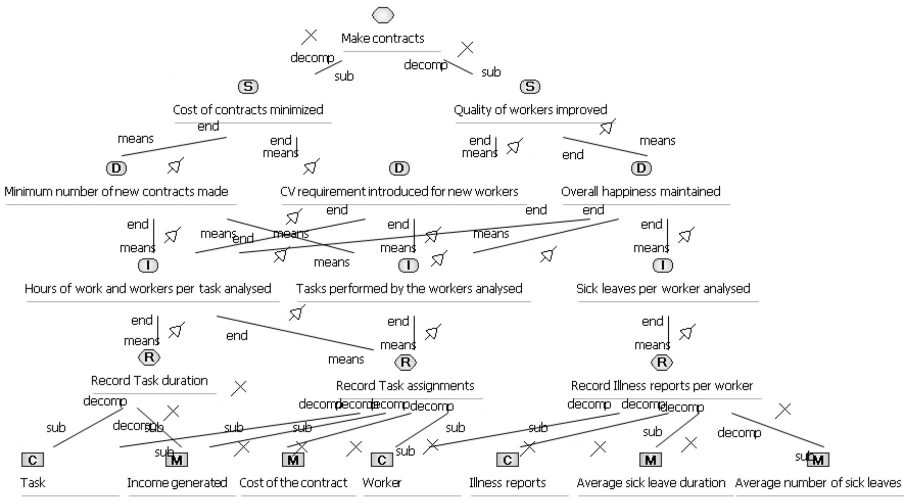

Make contracts
Cost of contracts minimized
Quality of workers improved
Minimum number of new contracts made
CV requirement introduced for new workers
Overall happiness maintained
Hours of work and workers per task analysed
Tasks performed by the workers analysed
Sick leaves per worker analysed
Record Task duration
Record Task assignments
Record Illness reports per worker
Task
Income generated
Cost of the contract
Worker
Illness reports
Average sick leave duration
Average number of sick leaves

Figure 1.11: Example of i* model for data warehouses.

objetivos tratan de responder a la pregunta "cómo se puede alcanzar un objetivo estratégico?", e intentan tomar las acciones adecuadas para cumplir con dicho objetivo. Los objetivos decisionales se encuentran relacionados con los objetivos estratégicos por medio de relaciones intencionales means-end. En nuestro ejemplo, con el fin de lograr *Cost of contracts minimized*, se ha decidido que es necesario tener el *Minimum number of new contracts made*, así como tener un *CV requirement introduced for new workers*, con el fin de lograr el objetivo estratégico. Sin embargo, los objetivos decisionales pueden afectar a más de un objetivo estratégico. En nuestro caso, el último objetivo decisional se encuentra relacionado también con el objetivo estratégico *Quality of workers increased*, ya que el currículum afecta a la calidad de los nuevos trabajadores que se emplea. Otros ejemplos de objetivos decisionales serían *Determine some kind of promotion* o *Open new stores*. Su cumplimiento sólo causa un beneficio para la organización si ayuda a alcanzar metas estratégicas, ya que los objetivos decisionales sólo se llevan a cabo dentro del contexto de los objetivos estratégicos.

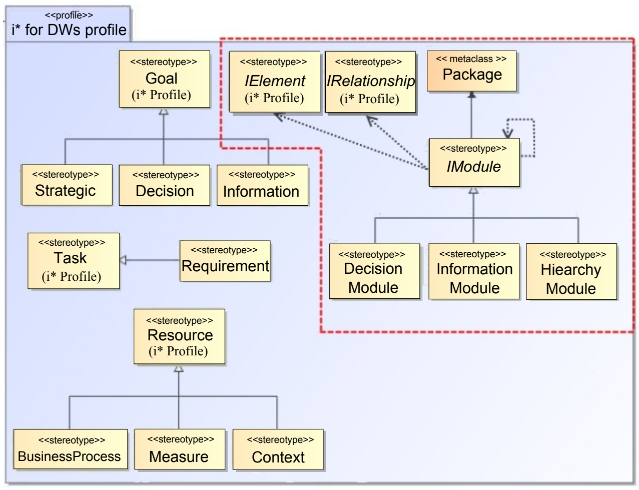
As with the strategic goals, the decision goals can be achieved by having the necessary information available. This required information is modeled by means of the informational goals. Information goals represent the lowest level of abstraction. They try to answer the question: "how can decision goals be achieved in terms of information required?", and they are related to the information required by a decision goal to be achieved. In our example, the information required is *Hours of work and workers per task analysed* and *Tasks performed by the workers analysed* for each decision goal, whereas the information about *Sick leaves per worker analysed* affects only the *Overall happiness maintained* decision goal. Other examples of information goals are *Analyze customer purchases* or *Examine stocks.* Their fulfillment helps to achieve decision goals and they only happen within the context of decision goals.

Como en el caso de los objetivos estratégicos, los objetivos decisionales se pueden alcanzar teniendo la información necesaria disponible. Esta información requerida se modela por medio de los objetivos de información. Los objetivos de información informáticos representan el nivel más bajo de abstracción. Tratan de responder a la pregunta: "cómo pueden lograrse los objetivos decisionales en términos de la información necesaria?", y están relacionados con la información requerida por un objetivo decisional que desea alcanzarse. En nuestro ejemplo, la información requerida es *Hours of work and workers per task analysed* y *Tasks performed by the workers analysed* para cada uno de los objetivo decisionales, mientras que la información acerca de *Sick leaves per worker analysed* afecta sólo al objetivo decisional *Overall happiness maintained.* Otros ejemplos de objetivos de información son *Analyze customer purchases* o *Examine stocks.* Su cumplimiento ayuda a lograr los objetivos decisionales y sólo ocurriren en el contexto de los objetivos decisionales.

Finalmente, los objetivos de información son alcanzados mediante requisitos de información. En nuestro caso, necesitamos realizar *Record Task duration*, *Record Task assignments*, y *Record Illness reports per worker* para disponer tener la información requerida. Cada uno de estos requisitos se descompone en contextos y medidas, que representan la información que se almacenará en el almacén de datos. Nuestro ejemplo incluye los contextos *Task, Worker*, e *Illness report*, así como las medidas *Income generated, Average sick leave duration* y *Average number of sick leaves*, que determinan el rendimiento de los procesos de negocio.

Como hemos visto, los modelos de requisitos van desde el nivel más alto de abstracción hasta los niveles más bajos. Los pasos básicos del proceso de elicitación requisitos son los siguientes: primero, el proceso comienza con la identificación de un proceso de negocio objetivo que el tomador de decisiones desea mejorar. Entonces, el tomador de decisiones describen los objetivos que quiere cumplir con el objetivo de mejorar los procesos de negocio. Estos objetivos estratégicos son a continuación refinados en los objetivos decisionales y de información, que representan las decisiones que se deben tomar y la información que se debe obtener. Este refinado se consigue mediante las preguntas "cómo?" y "por qué?" para ayudar en la exploración. Por último, se identifican los requisitos de información necesarios que den soporte a los objetivos de informaciónn. Estos requisitos de información agrupan las distintas entidades necesarias para alcanzar un objetivo de información.

Como podemos observar, los objetivos de nivel más bajo de abstracción sólo aparecen en el contexto de uno o varios objetivos de mayor abstracción.



<<profile>>
i* for DWs profile
<<stereotype>> Goal (i* Profile)
<<stereotype>> IElement (i* Profile)
<<stereotype>> IRelationship (i* Profile)
<< metaclass >> Package
<<stereotype>> IModule
<<stereotype>> Strategic
<<stereotype>> Decision
<<stereotype>> Information
<<stereotype>> Task (i* Profile)
<<stereotype>> Requirement
<<stereotype>> Decision Module
<<stereotype>> Information Module
<<stereotype>> Hiearchy Module
<<stereotype>> Resource (i* Profile)
<<stereotype>> BusinessProcess
<<stereotype>> Measure
<<stereotype>> Context

Figure 1.12: i* profile with modules extension for DW

Además, podemos observar una diferenciación entre la lógica del árbol de objetivos y las entidades de información que apoyan a estos objetivos. Con estas consideraciones, nuestra propuesta modularización está dirigido a (i) permitir una partición semántica de los diagramas de requisitos de los usuarios y (ii) separar los intereses empresariales (objetivos) de las entidades que se van a capturar en el almacén de datos. De esta manera, el diseñador puede concentrarse refinado de los modelos de objetivos en busca de objetivos adicionales o mejorar el detalle de las jerarquías y las medidas que se incluirán en el almacén de datos.

Los módulos definidos pueden verse en la Figura 1.12, donde extienden del concepto *Package* e incluyen una clase abstracta *iModule*, incluida con la finalidad de ayudar en la definición de restricciones OCL que garantizen su correcta aplicación.

- **Decision modules** incluyen los elementos relacionados con un *objetivo decisional.* Pueden incluir *objetivos decisionales, objetivos de información, requisitos, contextos, medidas y otros módulos de decisión, información,* y *módulos de jerarquía.* Contienen toda la información que es necesaria para tomar una decisión, la cual ayuda a la consecución de un *objetivo estratégico.*
- **Information modules** incluyen los elementos relacionados con un objetivo de información. Pueden incluir *objetivos de información, requisitos, contextos, medidas, otros módulos información,* y *módulos de jerarquía.* Estos módulos agrupan toda la información necesaria para satisfacer un

objetivo de información en concreto. *Nota: Este módulo fue transformado en un Information Requirements module después de un análisis detallado de los resultados del experimento.*

- **Hierarchy modules** incluyen los elementos que constituyen una jerarquía. Están compuestos por los diferentes contextos que representan los distintos niveles de agregación de una dimensión. Sólo pueden incluir *contextos*. Estos módulos ayudan con la reutilización de dimensiones a nivel de requisitos y ocultan la complejidad de las jerarquías cuando no resultan necesarias.

Además de la definición de estos módulos, se proponen una serie de recomendaciones para ayudar en su aplicación. La lsita completa de recomendaciones puede encontrarse en el Capítulo 5.

Después de definir el conjunto de módulos, realizamos un experimento con el fin de evaluar su idoneidad. Este experimento se realizó mediante la comparación de la eficacia y la calidad percibida de los modelos con y sin módulos al realizar varias tareas relacionadas con el modelado y la identificación de los requisitos de los usuarios. La descripción completa y los resultados del experimento pueden verse en el Capítulo 5. El análisis de los resultados muestra que (i) la tasa de errores se redujo en las tareas de identificación los elementos en los diagramas de tamaño moderado, y (ii) los participantes que utilizaban módulos olvidaban de forma sistemática las medidas del almacén cuando se le indicaba identificar los conceptos del diagrama relevantes para el almacén de datos subyacente. Teniendo en cuenta estos resultados, redujimos el nivel de abstracción de *Information modules*, y los sustituímos por *Informaquin requirements modules*, separando de forma efectiva los objetivos de los tomadores de decisión de las entidades capturadas en el almacén de datos.

Gracias a la definición de módulos semánticos hemos obtenido un metamodelo mejorado para los requisitos de usuario que permite a los diseñadores gestionar diagramas de una manera más fácil y mejora la comunicación con los tomadores de decisiones. Finalmente, en la siguiente sección, abordaremos el problema de la la brecha entre los diseñadores del almacén (expertos TI) y los tomadores de decisiones, la falta de una estrategia de Inteligencia de Negocio y los diferentes puntos de vista parciales proporcionados por los tomadores de decisiones individuales.

1.4.4 Alinemiento del Almacén de Datos con la Estrategia Corporativa

Incluso el almacén de datos mejor diseñado puede fallar a la hora de apoyar y mejorar el rendimiento del negocio, si (i) los tomadores de decisiones no son capaces de entender el significado de los datos y traducir la información a sus objetivos de negocio, o (ii) el almacén de datos no está alineado correctamente con la estrategia de negocio. Esta afirmación está respaldada por diversos estudios y encuestas, como [4], donde el Gartner Group destaca que una de las principales razones de la alta tasa de fracaso de los almacenes de datos es la brecha lingística entre la TI y los tomadores de decisiones.

Las aproximaciones recientes de desarrollo de almacenes de datos [25, 9] llegan hasta el punto de elicitar los requisitos de los usuarios en términos de objetivos de los tomadores de decisiones. Sin embargo, estudios anteriores han señalado que no es posible extraer un conjunto completo y preciso de requisitos de los tomadores de decisiones [41], ya que cada tomador de decisiones sólo puede proporcionar una visión parcial y personal. Por lo tanto, con el fin de hacer frente a los problemas anteriormente mencionados, en esta Tesis se propone enriquecer las técnicas de desarrollo de almacenes de datos actuales mediante la inclusión de la estrategia corporativa en el desarrollo de almacenes de datos.

El último problema al que enfrentan las aproximaciones actuales para el desarrollo de almacenes de datos es la validar los requisitos de usuario y asegurar que el almacén de datos datá el soporte adecuado a la estrategia del negocio. Los enfoques actuales [25, 9] recogen las necesidades de los tomadores de decisiones individuales, pero prestan poca atención a los objetivos de negocio. En el mejor de los casos, se realiza un modelado parcial de la actividad actual de la empresa [9] con el fin de ayudar en el proceso de reconciliación. Sin embargo, los objetivos de negocio, estrategias y planes son completamente ignorados.

Incluir el plan de la empresa en el proceso de desarrollo es una tarea difícil. La mayoría de la información disponible está escrita en lenguaje informal y, por lo tanto, no se puede utilizar directamente. Por lo tanto, el primer paso es formalizar el conocimiento incluido en el plan de la empresa. Para hacer frente a este problema, en esta Tesis se propone realizar este paso mediante el Business Motivation Model (BMM) [27]. BMM es un estándar propuesto por el Object Management Group que captura la semántica básica de los elementos que intervienen en un plan del negocio. Estos elementos van desde los objetivos hasta las iniciativas, incluyendo la Misión y la Visión de la organización. Sin embargo, estos elementos son sólo parcialmente formalizados en BMM. Cada elemento incluye sólo su tipo, un identificador y una descripción textual. Incluso las relaciones entre los elementos son opcionales. Por tanto, este nivel de abstracción limita la información que se puede extraer de estos elementos. Por ejemplo, teniendo en cuenta la meta[2] "To be a Premium Brand car rental company" y el objetivo "Be rated higher than 6 by AC Nielson in top car rental companies in EU", no hay manera de evaluar el tiempo restante para alcanzar la meta propuesta ni si esta meta se encuentra satisfecha o no de acuerdo al objetivo asociado, ya que no se dispone de los atributos *valor presente*, *valor objetivo* o *fórmula* que nos permita calcular esta información.

[2]En BMM se realiza una distinción entre *meta* y *objetivo*. El concepto de objetivo está más relacionado con el concepto de indicador.

Sin embargo, este inconveniente puede evitarse tal y como se muestra en el Capítulo 6 mediante la mejora de la formalización proporcionada por BMM. BMM puede servir como una representación intermedia para extraer la estructura inicial del plan del negocio. A continuación, se puede llevar a cabo un segundo paso de formalización, por ejemplo, mediante el modelo Business Intelligence Model (BIM) [3]. El modelo de BIM es un metamodelo diseñado para representar los objetivos y estrategias del negocio. A diferencia de i* para almacenes de datos, no se centra en los objetivos individuales de los tomadores de decisiones, sino en los objetivos de la empresa en su conjunto, y proporciona conceptos totalmente definidos para cada uno de los elementos incluidos en el modelo.

A continuación vamos a introducir los conceptos básicos mediante una breve descripción de un ejemplo de modelo BIM que se muestra en la Figura 1.13. En este ejemplo, la empresa de la EU-Rent desea alcanzar varios de objetivos de alto nivel a largo plazo denominados *Strategic Goals* y denotados como (SG). EU-Rent quiere posicionarse como "Premium Brand Car rental company" (SG1) y proveer "Industry-leading customer service" (SG2).

Los objetivos, y en especial los estratégicos, pueden tener *Indicators* que sirven para monitorizar su rendimiento. En este caso, podemos ver que EU-Rent dispone de dos indicadores para SG1. El primero, más específico, es "Be rated higher than 6 by AC Nielson in top car rental companies in EU" (xg1.1). El segundo, más general, es "Be rated higher than 9 by AC Nielson in top car rental companies" (xg1.2). Estos indicadores son utilizados por la compañía para analizar su rendimiento actual, y alertar acerca de posibles desviaciones sobre la planificación.

Los objetivos estratégicos se ven favorecidos por objetivos de nivel medio de abstracción y de medio plazo, los *Operational Goals*. Para alcanzar "Be Premium Brand car rental company" (SG1), es positivo si la empresa "Operate nation-wide in each country" (OG1). Estos objetivos operacionales se centran en contribuir a la satisfacción de los objetivos estratégicos y son descompuestos y soportados por *Tactical Goals*.

Un objetivo táctico representa un objetivo a corto plazo que puede ser operacionalizado directamente por los procesos de negocio de la empresa. Por ejempo, "Encourage rental extension" (TG1) puede ser parte de "Operate nation-wide in each country" (OG1) y operacionalizarse mediante el proceso de negocio "Car Rental". A pesar de que el metamodelo BIM permite añadir procesos de negocio de la empresa al modelo de la estrategia, no todos los planes del negocio llegan a este nivel de detalle, como es el caso del escenario EU-Rent scenario.

Finalmente, los objetivos pueden verse afectados por distintas *Situations* que pueden ayudar a su consecución o dificultarla. Una situación representa un análisis SWOT (Strengths, Weaknesses, Opportunities and Threats) [13] de los factores externos e internos que pueden afectar a la consecución de un objetivo. Por ejemplo, la presencia de “Budget Airlines” (S4) puede dificultar la consecución del objetivo “Operate nation-wide in each country” (OG1), ya que las aerolíneas de bajo coste son un fuerte competidor de las empresas de alquiler de coches para llevar a cabo viajes de larga distancia dentro de un mismo país.

Por medio de estos conceptos se puede formalizar aún más el plan del negocio, obteniendo como resultado una estrategia de negocio totalmente formalizada. Una vez formalizada la información almacenada en el plan del negocio, podemos incluir esta información en el proceso de desarrollo del almacén de datos, con el fin de validar los requisitos de los usuarios.

Para llevar a cabo esta tarea, seguimos el enfoque descrito en el Capítulo 6, que consiste en los pasos descritos a continuación. En primer lugar, se realiza un análisis de requisitos inicial para capturar los requisitos de los distintos tomadores de decisiones. En nuestro caso, llevamos a cabo este paso mediante el uso de nuestro i* para almacenes de datos mejorado [23]. Posteriormente, se obtiene un modelo de estrategia empresarial a partir del plan de la empresa que incluye los objetivos del negocio. Una vez que hemos recogido tanto los requisitos de los usuarios y el modelo de la estrategia de negocio, alineamos los objetivos de los diferentes tomadores de decisiones con el modelo de la estrategia de negocio, tal y como se ejemplifica en la Figura 1.14. Para llevar a cabo este alineamiento es necesario (i) aplicar una serie de restricciones que se describen en el Capítulo 6, y (ii) involucrar a un experto en el dominio con el fin de garantizar una alineación correcta. Por último, se procede a analizar los resultados.

Como resultado del alineamiento, podemos identificar qué objetivos de negocio están siendo considerados por los tomadores de decisiones. Por ejemplo,

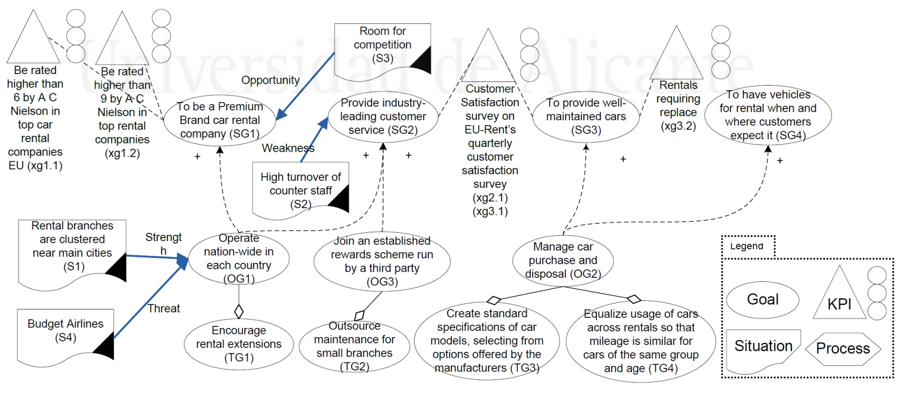

Be rated higher than 6 by A C Nielson in top car rental companies EU (xg1.1)
Be rated higher than 9 by A C Nielson in top rental companies (xg1.2)
Opportunity
Room for competition (S3)
To be a Premium Brand car rental company (SG1)
Provide industry-leading customer service (SG2)
Weakness
High turnover of counter staff (S2)
Customer Satisfaction survey on EU-Rent's quarterly customer satisfaction survey (xg2.1) (xg3.1)
To provide well-maintained cars (SG3)
Rentals requiring replace (xg3.2)
To have vehicles for rental when and where customers expect it (SG4)
+
Rental branches are clustered near main cities (S1)
Strength
Operate nation-wide in each country (OG1)
Join an established rewards scheme run by a third party (OG3)
Manage car purchase and disposal (OG2)
Budget Airlines (S4)
Threat
Encourage rental extensions (TG1)
Outsource maintenance for small branches (TG2)
Create standard specifications of car models, selecting from options offered by the manufacturers (TG3)
Equalize usage of cars across rentals so that mileage is similar for cars of the same group and age (TG4)
Legend
Goal
KPI
Situation
Process

Figure 1.13: Example business strategy modeled after the EU-Rent scenario.

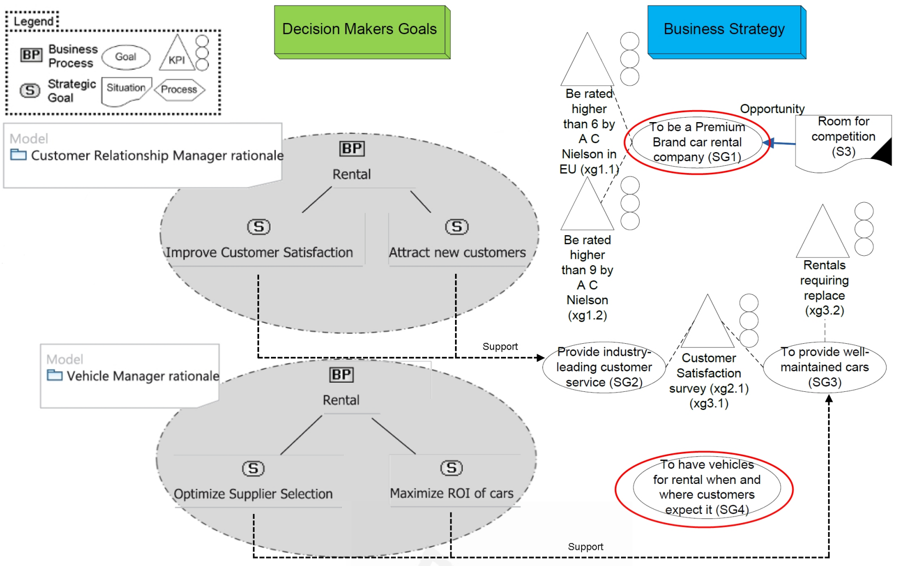

Legend
BP Business Process
Goal
KPI
S Strategic Goal
Situation
Process
Decision Makers Goals
Business Strategy
Model
Customer Relationship Manager rationale
BP Rental
S Improve Customer Satisfaction
S Attract new customers
Model
Vehicle Manager rationale
BP Rental
S Optimize Supplier Selection
S Maximize ROI of cars
Support
Support
Be rated higher than 6 by A C Nielson in EU (xg1.1)
Be rated higher than 9 by A C Nielson (xg1.2)
Opportunity
To be a Premium Brand car rental company (SG1)
Room for competition (S3)
Rentals requiring replace (xg3.2)
Provide industry-leading customer service (SG2)
Customer Satisfaction survey (xg2.1) (xg3.1)
To provide well-maintained cars (SG3)
To have vehicles for rental when and where customers expect it (SG4)

Figure 1.14: Alignment between decision maker goals and the overall strategy.

en la Figura 2.14, podemos ver como el tomador de decisiones “Customer Relationship Manager” se está centrando en el objetivo “Provide industry-leading customer service” (SG2), y el tomador de decisiones “Vehicle Manager” está trabajando de forma activa en el objetivo de negocio “Provide well-maintained cars” (SG3). Por tanto, decimos que SG2 y SG3 están actualmente siendo *Soportadas* por el almacén de datos y por el proceso de toma de decisiones. Además, también podemos identificar que ningún tomador de decisiones tiene objetivos personales alineados con “To be a Premium Brand car rental company” (SG1) y “To have vehicles for rental when and where customers expect it” (SG4). Por lo tanto, decimos que SG1 y SG4 son *No soportadas* por el almacén de datos y por el proceso de toma de decisiones. Es decir, en esta situación, el almacén de datos puede carecer de la información requerida para tomar decisiones acerca de estos objetivos del negocio y, por tanto, debería de ser analizada e incluida. Además, podemos ver que no hay dos tomadores de decisiones trabajando en el contexto del mismo objetivo del negocio, por lo que, inicialmente, no hay información compartida o colaboración potencial. Finalmente, podemos ver que no hay ningún objetivo particular que esté desalineado con la estrategia del negocio, por lo que podemos decir que los requisitos de los usuarios son válidos desde el punto de vista del negocio y no es necesaria ninguna corrección.

1.4.5 Discusión y Conclusiones

En esta tesis doctoral se han analizado las dificultades actuales en el desarrollo de almacenes de datos y se han presentado las bases para un enfoque de desarrollo de almacenes de datos mejorado que supera estas dificultades. El conjunto de herramientas presentadas incluye un metamodelo de trazabilidad que permite mantener la trazabilidad durante todo el proceso de desarrollo, una formalización para que el diseñador del almacén pueda documentar y derivar un almacén de datos reconciliado, una propuesta de modularización para mejorar la comunicación con los usuarios utilizando modelos i* basados en objetivos, y un método de alineamiento para validar los requisitos de los usuarios y asegurar que el almacén de datos está alineado con el plan del negocio. Además, hemos integrado estas propuestas en una aproximación para el desarrollo de almacenes de datos utilizando [25] como base. Finalmente, cabe destacar que esta propuesta se puede combinar con la mayoría de aproximaciones de desarrollo de almacenes de datos existentes, permitiendo así flexibilidad en la elección del conjunto los modelos utilizados.

Los resultados obtenidos en esta tesis doctoral son:

- La preservación de la trazabilidad en el desarrollo del almacén de datos.
- La formalización y documentación del proceso de reconciliación.
- Mejora del mantenimiento del almacén de datos y sus modelos asociados.
- Mejora de la comunicación con los usuarios por medio de diagramas particionados.
- El alineamiento de los requisitos de los usuarios y el almacén de datos con el plan del negocio.

Sin embargo, todavía hay margen de mejora. En primer lugar, se podría desarrollar una aproximación de identificación automática de correspondencias, con el fin de ayudar al diseñador en la adecuación de las fuentes de datos con los modelos multidimensionales definidos a partir de los requisitos de los usuarios. En segundo lugar, la extensión hasta donde los modelos estratégicos de negocio pueden ser explotados es aún desconocida. Del mismo modo, la información de trazabilidad permite el uso de técnicas que no podrían ser utilizadas previamente o eran altamente costosas en términos de recursos, por lo que esto podría conducir a extensiones adicionales del modelo de trazabilidad y a la creación de nuevos algoritmos.

Como resultado, el trabajo realizado en esta Tesis Doctoral abre nuevas líneas de investigación.

1. En primer lugar, una línea de análisis de la relación entre los modelos de estrategia empresarial y el almacén de datos. Cómo el almacén de datos

puede dar soporte más eficaz a los planes del negocio y a su evolución y, a la inversa, cómo los cambios en la estrategia de negocio afectan al almacén de datos.

2. En segundo lugar, una vez que se ha identificado que varios tomadores de decisiones se encuentran trabajando en el contexto del mismo objetivo de negocio, cómo pueden colaborar e intercambiar información con el fin de lograr un mayor éxito.

3. En tercer lugar, dado el creciente número de fuentes de datos disponibles a nivel mundial, un tema interesante es analizar cómo los almacenes de datos pueden hacerse más flexibles a la incorporación de nuevas fuentes de datos y cómo pueden relacionarse estas fuentes de datos con las estructuras existentes en el esquema del almacén de forma semi-automática.

4. Por último, en cuarto lugar, el soporte de trazabilidad presentado en esta Tesis permite acceder a todos los elementos relacionados con un requisito dado a distintos niveles de abstracción. Por lo tanto, se puede definir y calcular un nuevo conjunto de medidas automáticas que guíen al diseñador en el proceso de diseño e implementación del almacén de datos. Tal conjunto de medidas podría no sólo analizar la calidad del almacén de datos, sino que también podría incluir estimaciones acerca del esfuerzo requerido para completar la implementación.

Bibliography

[1] N. Aizenbud-Reshef et al. "Model traceability". In: *IBM Systems Journal* 45.3 (2006), pp. 515–526.

[2] G. Antoniol et al. "Recovering traceability links between code and documentation". In: *IEEE Transactions on Software Engineering* 28.10 (2002), pp. 970–983.

[3] D. Barone, T. Topaloglou, and J. Mylopoulos. "Business intelligence modeling in action: a hospital case study". In: *Advanced Information Systems Engineering.* Springer. 2012, pp. 502–517.

[4] A. Bitterer, K. Schlegel, and D. Laney. *Predicts 2012: Business Intelligence Still Subject to Nontechnical Challenges.* 2011. URL: http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1873915.

[5] A. Bonifati et al. "Designing data marts for data warehouses". In: *ACM Transactions on Software Engineering and Methodology* 10.4 (2001), pp. 452–483.

[6] M.D. Del Fabro, J. Bézivin, and P. Valduriez. "Weaving Models with the Eclipse AMW plugin". In: *Eclipse Modeling Symposium, Eclipse Summit Europe 2006.* Esslingen, Germany. 2006.

[7] X. Franch. "Incorporating Modules into the i* Framework". In: *CAiSE.* Vol. 6051. LNCS. Springer Berlin, 2010, pp. 439–454.

[8] X. Franch, A. Maté, and J. Trujillo. "On the joint use of i* with other Modelling Frameworks: a Vision Paper". In: *Proceedings of the 19th International Conference on Requirements Engineering.* IEEE, In Press.

[9] P. Giorgini, S. Rizzi, and M. Garzetti. "GRAnD: A goal-oriented approach to requirement analysis in data warehouses". In: *Decision Support Systems* 45.1 (2008), pp. 4 –21.

[10] M. Golfarelli, D. Maio, and S. Rizzi. “The dimensional fact model: a conceptual model for data warehouses”. In: *International Journal of Cooperative Information Systems* 7.2 (1998), pp. 215–247.

[11] O. Gotel and S. Morris. “Macro-level Traceability Via Media Transformations”. In: *Requirements Engineering: Foundation for Software Quality*. Vol. 5025. Lecture Notes in Computer Science. Springer Berlin, 2008, pp. 129–134.

[12] Orlena CZ Gotel and CW Finkelstein. “An analysis of the requirements traceability problem”. In: *Proceedings of the First International Conference on Requirements Engineering*. IEEE. 1994, pp. 94–101.

[13] T. Hill and R. Westbrook. “SWOT analysis: it’s time for a product recall”. In: *Long Range Planning* 30.1 (1997), pp. 46–52.

[14] B. Hüsemann, J. Lechtenbörger, and G. Vossen. “Conceptual data warehouse design”. In: *Proc. DMDW*. 2000, pp. 3–9.

[15] W.H. Inmon. *Building the data warehouse*. Wiley-India, 2009.

[16] F. Jouault. “Loosely coupled traceability for atl”. In: *ECMDA-TW*. Nuremberg, Germany. 2005, pp. 29–37.

[17] F. Jouault and I. Kurtev. “Transforming models with ATL”. In: *Satellite Events at the MoDELS 2005 Conference*. Springer. 2006, pp. 128–138.

[18] R. Kimball et al. *The data warehouse lifecycle toolkit*. Wiley, 2011.

[19] A.G. Kleppe, J. Warmer, and W. Bast. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Longman Publishing, 2003.

[20] James Kobielus et al. “Mighty mashups: do-it-yourself business intelligence for the new economy”. In: *Forrester Research* (2009).

[21] S. Luján-Mora, J. Trujillo, and I.Y. Song. “A UML profile for multidimensional modeling in data warehouses”. In: *Data & Knowledge Engineering* 59.3 (2006), pp. 725–769.

[22] A. Maté and J. Trujillo. “A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses”. In: *Information Systems* 37.8 (2012), pp. 753 –766.

[23] A. Maté, J. Trujillo, and X. Franch. “Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star”. In: *Journal of Systems and Software (In Press)* ().

[24] J.-N. Mazón, J. Pardillo, and J. Trujillo. “A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses”. In: *Advances in Conceptual Modeling Foundations and Applications*. Vol. 4802. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, pp. 255–264.

[25] J-N. Mazón and J. Trujillo. “An MDA approach for the development of data warehouses”. In: *Decision Support Systems* 45.1 (2008), pp. 41–58.

[26] J.-N. Mazón, J. Trujillo, and J. Lechtenbörger. “Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms”. In: *Data & Knowledge Engineering* 63.3 (2007), pp. 725–751.

[27] Object Management Group. *Business Motivation Model (BMM).* 2010.

[28] Object Management Group. *The Meta Object Facility 2.0 Query/View-/Transformation.* 2005.

[29] R.F. Paige et al. “Building model-driven engineering traceability classifications”. In: *ECMDA-TW* (2008), pp. 49–58.

[30] N. Prakash and A. Gosain. “Requirements driven data warehouse development”. In: *CAiSE Short Paper Proceedings.* 2003, pp. 13–17.

[31] B. Ramesh and M. Jarke. “Toward reference models for requirements traceability”. In: *IEEE Transactions on Software Engineering* 27.1 (2001), pp. 58–93.

[32] O. Romero and A. Abelló. “A framework for multidimensional design of data warehouses from ontologies”. In: *Data & Knowledge Engineering* 69.11 (2010), pp. 1138–1157.

[33] C. Sapia et al. “Extending the E/R Model for the Multidimensional Paradigm”. In: *Proceedings of the Workshops on Data Warehousing and Data Mining: Advances in Database Technologies.* Springer-Verlag. 1998, pp. 105–116.

[34] A. Sen and A.P. Sinha. “A comparison of data warehousing methodologies”. In: *Communications of the ACM* 48.3 (2005), pp. 79–84.

[35] Alkis Simitsis, Panos Vassiliadis, and Timos K Sellis. *Extraction-Transformation-Loading Processes.* 2005.

[36] G. Spanoudakis and A. Zisman. “Software traceability: a roadmap”. In: *Handbook of Software Engineering and Knowledge Engineering* (2005).

[37] N. Tryfona, F. Busborg, and J.G. Borch Christiansen. “starER: A conceptual model for data warehouse design”. In: *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP.* ACM. 1999, pp. 3–8.

[38] P. Vassiliadis. “Data Warehouse Modeling and Quality Issues”. PhD thesis. Athens, 2000.

[39] S. Walderhaug et al. “Traceability in Model-Driven Software Development”. In: *Designing Software-Intensive Systems: Methods and Principle* (2008), pp. 133–159.

[40] S. Winkler and J. von Pilgrim. “A survey of traceability in requirements engineering and model-driven development”. In: *Software and Systems Modeling* 9 (4 2010), pp. 529–565.

[41] R. Winter and B. Strauch. “A method for demand-driven information requirements analysis in data warehousing projects”. In: *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on.* IEEE. 2003.

[42] E. Yu. “Modeling strategic relationships for process reengineering”. In: *Social Modeling for Requirements engineering* (2011).

[43] Y. Yu, J. Jurjens, and J. Mylopoulos. “Traceability for the maintenance of secure software”. In: *IEEE International Conference on Software Maintenance (ICSM 2008)*. IEEE. 2008, pp. 297–306.

2

Summary in English

This PhD Thesis is presented as a collection of published papers. Thus, as required by the University of Alicante, this chapter provides a description and a summary of the initial hypotheses, research objectives, and the collection of works that comprise this PhD Thesis. In order to achieve this goal, this chapter justifies and provides a summary of the scientific content included in each chapter of this PhD Thesis, including motivations, research done, and final conclusions.

2.1 Research Objectives and Initial Hypotheses

In the early 90's data warehousing techniques are proposed to store massive, historical data of the organization [15, 18]. Data warehousing aims to support the decision making process by obtaining data from several, heterogeneous data sources and integrating them into a repository for the organization. These data are then periodically refreshed with the newest data available from each data source by means of Extraction/Transformation/Load (ETL) processes [35].

Data warehouse development is a long-term, costly task, which may require years to be built depending on the size of the data warehouse. Therefore, careful design and maintenance is required in order to be successful. Initial approaches proposed to design a data warehouse are supply-driven, and only consider the information stored in the data sources [15, 18]. These approaches start by modeling the multidimensional concepts, i.e. facts and dimensions, available from the data sources. A fact stores measures related to the performance of a business process (e.g. quantity of products sold) while a dimension stores relevant context information (e.g. product data). After identifying the multidimensional concepts, these approaches design a star schema [18] at the logical level in order to create the tables that will store the data.

Given the absence of any formalism to model the multidimensional aspects of the data warehouse, several works have been proposed to define a conceptual data warehouse model [21, 37, 10, 14, 33]. However, even when including conceptual modeling, supply-driven approaches ignore user requirements until the data warehouse has already been built. This may lead to user insatisfaction and an inappropriate support of the decision making process [9].

In order to solve this drawback, demand-driven and hybrid approaches are proposed [34, 9, 25, 18, 30, 5]. On the one hand, demand-driven approaches [41, 5, 30] introduce an initial requirement analysis step in the process. They focus on building a data warehouse by paying particular attention to user requirements. Then, once the data warehouse has been implemented, data is loaded from the different data sources.

However, it is rarely the case that data warehouse users, who are mostly business people, have accurate and comprehensive knowledge of the operational data sources from where information is extracted, as they are not IT experts [41]. Therefore, it is not guaranteed that (i) all the information required is stored, or in the expected format, and (ii) important information has not been overlooked during the requirements phase. On the other hand, hybrid approaches [9, 25] include a data source analysis step before implementing the data warehouse. Thus, these approaches are able to identify earlier which requirements cannot be satisfied and what relevant data may have been overlooked.

Nevertheless, even the most recent data warehouse development approaches focus on providing the tools and steps for designing the data warehouse, but little attention is paid to other aspects such as:

1. The lack of a long-term Business Intelligence plan that includes the business goals. Since requirements are elicitated from individual decision makers, they provide partial views that may not be completely aligned with business goals. Additionally, the different language employed by decision makers and data warehouse designers combined with partial views of the requirements makes difficult to validate user requirements in early stages and has been reported to be one of the major causes of failure [4].

2. Ensuring that the resulting data warehouse multidimensional models adequately fulfill the specified user requirements. So far, only the work in [38], on which [5] is also based, proposes measures to evaluate the quality of the data warehouse multidimensional schema with regards to

requirements. However, some of the measures proposed can be difficult to calculate since they require information from several models where the same elements may have different names or even structure and must be manually traced.

3. Lowering the effort required to manage several models involved in data warehouse development. As data warehouses involve several models with different abstraction levels, (requirements, conceptual, logical and physical), tracking down an error or finding the stakeholder, or data source, responsible for the structure of certain cubes can be a challenging task.

4. Supporting the evolution and maintenance of the data warehouse once it has been implemented. Providing tools that allow the designer to identify the scope of a change can be crucial in order to avoid undesired effects in other elements of the business intelligence system, such as dashboards or scorecards.

Therefore, all these aspects are left to the expertise of the data warehouse designer, and even in the case of experts it is not guaranteed that the data warehouse project will be successful. Thus, the **hypothesis of this PhD Thesis** is that data warehouse development can be improved by:

1. Incorporating traceability in the development process, allowing to automate requirements validation by using the data warehouse schema and supporting the management of changes as the data warehouse evolves, thus saving development time and avoiding errors.

2. Enabling the calculus of a series of measures over complex data warehouse models which provide an idea of how well the proposed design fits user requirements.

3. Aligning the development of the data warehouse with the business goals behind the decisions taken by the decision makers, in order to ensure that the data warehouse built supports the business objectives.

So far, despite previous efforts, data warehouse projects still have a rate of failure higher than 70% [4] and present a gap between IT and business people which limits the benefits obtained from the data warehouse. Thus, in order to develop our proposal, we build on top of the work developed in [25] within the Lucentia Research Group, in which a hybrid Model Driven approach based on the Model Driven Architecture (MDA) [19] is proposed to develop data warehouses.

To conclude, **the research objective of this PhD Thesis** is to define a set of techniques and approaches in order to improve the data warehouse development process by tackling the existing problems by (i) introducing and preserving traceability during the development process, (ii) formalizing the relationships involved in the reconciliation process, and (iii) aligning data warehouse development with the business strategy model, thus guaranteeing that the data warehouse supports the business goals, and, as a result, improving the success rate of data warehouse projects.

2.2 List of Publications included in this PhD Thesis

This section presents a collection of papers which have been chosen to be part of this PhD Thesis due to their relevance and contribution. Each of this papers is briefly described in this section, including how it relates to the PhD Thesis and the chapter where it is included.

Chapter 3

***Maté, A.** and Trujillo, J. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. Information Systems, (**IS**). 2012. Impact factor: 1.595*

Traceability is a relevant topic in data warehouses since data warehouses are complex structures that are in constant evolution. Whenever a requirement or data source is added or changed, it has certain impact on the data warehouse. Most changes will incur into a loss of the implicit traceability between the different data warehouse models, thus hindering the ability of the designer to validate resulting data warehouse. Moreover, as data warehouses are one of the core elements in current business intelligence systems, any delays and errors in the process will affect the decision making ability of the enterprise.

In this initial chapter we analyze the support for traceability provided in current data warehouse development approaches. Our analysis shows that current development approaches assume the existence of traceability as implicit knowledge, rather than explicitly modeling it. However, traditionally explicit traceability incurs into an overhead cost derived from creating and maintaining traces. Given that data warehouse development is already a long and costly process, any overhead introduced must be minimized. In order to avoid this problem, in this work we describe a traceability proposal that can be integrated into current data warehouse development approaches and covers the whole set of models and relationships involved in the process. Our proposal allows the designer to accurately trace any requirement to the data warehouse and back, thus being able to adequately validate the requirements and accurately assess the impact of changes. Furthermore, as is shown, when included within a Model Driven Development framework our proposal introduces no overhead in the data warehouse development process and, in addition, it can automate different tasks such as impact analysis, change propagation and requirements validation.

Chapter 4

Maté, A. *and Trujillo, J. Incorporating Traceability in Conceptual Models for Data Warehouses MDA. Proceedings of 30th International Conference on Conceptual Modeling (**ER'11**). 2011. Brussels, Belgium. Acceptance rate: 24.8%. ERA A*

Maté, A. *and Trujillo, J. and De Gregorio, E. and Song, I. Y. Improving the Maintainability of Data Warehouse Designs: Modeling Relationships between Sources and User Concepts. Proceedings of 15th Workshop on Data Warehousing and OLAP, (**DOLAP 2012**). 2012. Maui, Hawaii. Acceptance rate: 30%. ERA B*

The analysis performed in the previous chapter shows that the data integration and reconciliation process in data warehouses is carefully considered by the designer while modeling the data warehouse following a supply-driven approach or a hybrid approach. However, no modeling tools are provided to model this integration within the current data warehouse development frameworks. Instead, it is first analyzed while the data warehouse is being developed and, then, modeled separately in Extraction/Transformation/Load design tools after the data warehouse has already been implemented.

Nevertheless, data integration flows represent a crucial part of the data warehouse and the business intelligence system structure. Any changes in the data flows are immediately translated into the data warehouse, affecting several requirements, reports, and dashboards. For example, changing a source column may affect over 40 different reports, dashboards, and other decision maker interfaces [20]. Thus, it is important to be able to accurately identify the effect of each change in the data warehouse and reacommodate the data warehouse implementation. Therefore, in this chapter, we first analyze how the relationships between requirements and data source structures affect the resulting data warehouse as well as its requirements. We identify two potential types of relationships *Overlap* and *Conflict* that affect how the contextual information of the data warehouse (i.e. dimensions) is derived. On the one hand, an overlap represents a match in the structure of the dimension and its hierarchy between the requirements and the data source. On the other hand, a conflict represents a mismatch in this structure, and highlights the need to consider and decide about what will be the final structure of the dimension. According to these relationships, we define a set of Query/View/Transformation rules to derive the final data warehouse schema.

Second, we propose a formalization to model the relationships identified during the reconciliation process between the data warehouse and the data sources. Our formalization extends our previous overlap and conflict relationships into a more detailed set of traces. This approach allows the designer to model and document all the information identified during the reconciliation process, thus providing accurate information and making it easier to (i) integrate new data sources, (ii) analyze the degree of satisfaction of user requirements, and (iii) specify ETL processes. In turn, we improve the maintainability of the data warehouse and avoid repetitive, error-prone tasks, such as inspecting the different data and merge the design of the data warehouse with data source information, that would be required to be performed whenever a change is made.

Chapter 5

Maté, A. *and Trujillo, J. and Franch, X. Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star. Journal of Systems and Software (**JSS**). (In Press) Impact Factor: 1.135*

The previous chapters tackle the challenges related to data warehouse development complexity and data warehouse validation by providing a traceability scaffolding throughout the different models involved in data warehouse development. However, we still require to tackle the problem of validating requirements in data warehouses. This problem will be tackled in two parts. First, we improve the requirements models used, which is the focus of this chapter. Second, we will tackle the problem of partial requirement views and validating user requirements in the next chapter.

Most of the recent data warehouse development approaches make use of goal-based models for elicitating and modeling requirements. Among goal models, one of the most popular ones that has been applied to different areas, including data warehouses, has been i* [42]. Goal models present the advantage that are easier to understand by business people. However, i* presents a scalability problem when models become too big since it lacks any modularization mechanisms [7]. In turn, this hinders requirements validation, as diagrams become harder to use for communication between designers and users.

In this chapter, we analyze the effectivity of current i* based requirements modeling proposals for data warehouses. First, we tackle the problem of scalability by adding a set of semantic modules to the metamodel [23, 8], thus allowing the designer to better partition and present the models to the users. Afterwards, we perform an experiment to compare the effectivity of the models with and without modules. The results show a trend that diminishes the error rate as the requirements models get bigger, as well as an increase in the number of elements identified while modeling data warehouse requirements.

Chapter 6

Maté, A. *and Trujillo, J. and Yu, E. Aligning Data Warehouse Requirements with Business Goals. Proceedings of the Sixth International i* Workshop (iStar 2013), CEUR Vol. 978, pp. 67-72. 2013 Valencia, Spain.*

Once the requirements models have been improved in order to facilitate the communication with users, we focus on the last challenge of tackling the problem of validating user requirements.

Recent data warehouses development approaches [25, 9] elicitate requirements from decision makers in order to ensure that the data warehouse supports the decision making process. However, these approaches (i) cover only the goals of individual decision makers, and (ii) only make use of business goal modeling to aid in the identification of relevant information to be included in the data warehouse. Thus, the requirements elicitated (i) provide only partial views of the system and (ii) do not contribute to lessen the gap between IT and decision makers, since designers cannot adequately interpret requirements in business terms [4]. Thus, in this chapter we go one step further than current state of the art approaches and we (i) incorporate the information from the business plan into data warehouse requirements validation, and (ii) connect the data warehouse with a business strategy model.

In this chapter we define a process to validate each user requirement against the information stored in the business plan. To this aim, we first represent the business plan using the Business Intelligence Model (BIM) [3]. Then, we define a set of constraints to that must be met in order for a user requirement to be aligned with the business strategy. The final result is a data warehouse that is aligned with the business strategy. Additionally, our proposal allows us to identify any goals that have been overlooked in the data warehouse system, as well as the different decision makers involved on taking decisions targeting the same business goals, thus allowing us to compare the information used by each one and provide a more complete and integrated solution.

Appendix A

Maté, A. *and Trujillo, J. Tracing Conceptual Models Evolution in Data Warehouses by using MDA. Computer Standards and Interfaces. Under review. (2nd round) Impact Factor: 0.978*

In this chapter we present an extension of our work for documenting the relationships between the data warehouse and the different data sources. This chapter includes a formalization of the basic trace types involved in the reconciliation process as well as an improved set of QVT relationships that allow to derive the final data warehouse schema from any trace configuration. Thus, we enable a quick reconfiguration and analysis of the data warehouse whenever a data source is changed or added.

2.3 Other publications in International conferences

This section covers a set of papers that have been published as part of research done during the PhD Thesis. However, these papers have not been included in this collection as they are complementary to the core of the PhD Thesis.

Maté, A. *and Trujillo, J. and Franch, X. A modularization proposal for goal-oriented analysis of data warehouses using i-star. Proceedings of 30th International Conference on Conceptual Modeling (**ER'11**). Lecture Notes in Computer Science Vol. 6998, pp. 421-428. 2011. Brussels, Belgium. Acceptance rate: 24.8%. ERA A*

Franch, X. and ***Maté, A.*** *and Trujillo, J. and Cares, C. On the joint use of i* with other modelling frameworks: A vision paper. Proceedings of the 19th IEEE International Requirements Engineering Conference (**RE**). 2011. Trento, Italy. Acceptance rate: 16,7%. ERA A*

Maté, A. *and Trujillo, J. and Mylopoulos, J. Conceptualizing and Specifying Key Performance Indicators in Business Strategy Models. Proceedings of 31th International Conference on Conceptual Modeling, (**ER'12**). Lecture Notes in Computer Science, Vol. 7532, pp. 282-291 2012. Florence, Italy. Acceptance rate: 26,2%. ERA A*

Trujillo, J. and ***Maté, A.*** *Business Intelligence 2.0: A General Overview. Lecture Notes in Business Information Processing Vol. 96, pp. 98-116. 2012. Springer.*

Maté, A. *and Llorens, H. and de Gregorio, E. An Integrated Multidimensional Modeling Approach to Access Big Data in Business Intelligence Platforms. Proceedings of the First International Workshop on Modeling for Data-Intensive Computing. Lecture Notes in Computer Science Vol. 7518, pp. 111-120. 2012 Florence, Italy.*

Maté, A. *and Trujillo, J. and Mylopoulos, J. Conceptualizing and Specifying Key Performance Indicators in Business Strategy Models. Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research (CASCON'12). pp. 102-115. 2012 Toronto, Canada*

Maté, A. *and de Gregorio, E. and Cámara, J. and Trujillo, J. Improving Massive Open Online Courses Analysis by applying Modeling and Text Mining: a Case Study.* Proceedings of the First International Workshop on Modeling and Management of Big Data. 2013 (In Press)

2.4 Summary of the PhD Thesis

The main objective of this PhD Thesis is to define a series of techniques in order to enhance data warehouse development approaches, thus covering the shortcomings of previous proposals. Data warehouses aim to support the decision making process by integrating several heterogeneous data sources into a single truth for the organization. Data warehouse development is a long, complex process which may take up years to complete. Despite such effort, data warehouses projects rarely meet user needs, failing over 70% of the time according to recent studies [4].

Given the special idiosyncrasy of data warehouses, a specific development methodology is required that differs from standard software and database development methodologies. There are several differences:

1. First, a data warehouse is conceptually modeled in terms of facts, center of the analysis, and dimensions, context of analysis.

2. Second, The structure of the data warehouse, in terms of facts and dimensions, usually depends on the information that decision makers wish to analyze. However, the data to be stored inside a data warehouse and its structure also depend on the availability of such information in the data sources, thus not all user requirements can be satisfied.

3. Third, fresh data is required in order to take decisions, thus the data warehouse must be kept up to date with any changes not only in user requirements, but also in the data sources from which data is loaded.

4. Finally, not all information can be stored into a single OLAP schema (often identified as an analysis cube), since (i) different users are interested in different information, and (ii) the complexity of the information shown increases as more information is added thus hurting the understandability of the data shown and impacting the performance of queries.

In order to address these specific characteristics, demand-driven, supply-driven, and hybrid approaches have been proposed in literature [34, 15, 9, 25, 18, 30, 5]. An overview of the steps involved in these approaches is shown in Figure 2.1. First, in both demand-driven and hybrid approaches, user requirements are gathered. Then, according to these requirements, an initial multidimensional schema is designed. In the case of pure demand-driven approaches, this schema is implemented and loaded with data. On the other hand, hybrid approaches perform a thorough inspection of data sources and then remodel the initial schema according to the information available. In the

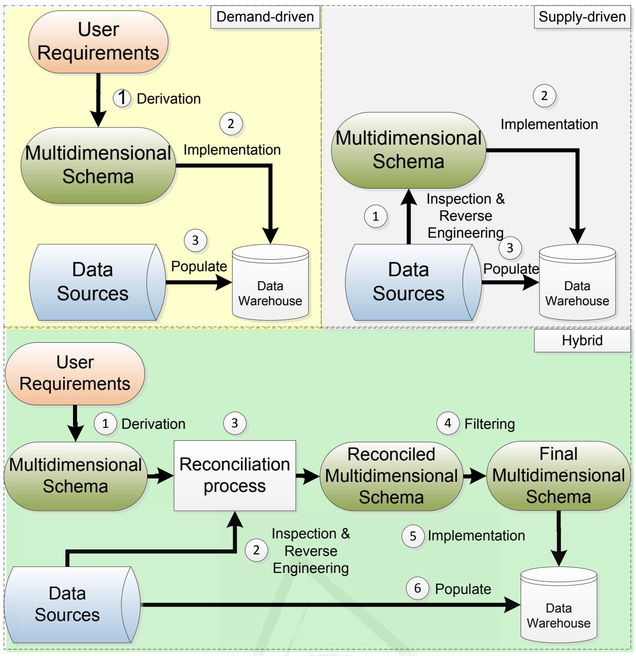


Figure 2.1: Overview of steps included in demand-driven, supply-driven and hybrid approaches.

case of pure supply-driven approaches, the inspection of data sources provides the concepts to be modeled in the multidimensional schema. Finally, the data warehouse according to this schema is implemented and loaded with data.

Each of the presented techniques has different drawbacks. Demand-driven approaches require that the data warehouse or data mart to be designed is relatively small and that there is detailed knowledge of the data stored in the data sources [9] in order to be successful. Supply-driven approaches mostly ignore user requirements, thus it is possible that decision makers find it difficult to understand the schema that they are analyzing. Additionally, in both approaches it is also possible that we are missing important information either because decision makers cannot provide a comprehensive list of all the information they need [41] or because, as we ignored user requirements, there are entities and attributes that should have been derived from existing information but where not. These drawbacks have led to an emergence of hybrid techniques in the recent years [25, 9]. While hybrid approaches solve the aforementioned problems, they incur into a loss of traceability derived from the reconciliation process as we will see in the following section.

2.4.1 A Traceability Metamodel for Tracing User Requirements

In order to adequately validate user requirements and estimate the suitability of the data warehouse proposed, it is necessary to be able to trace every requirement up to the final implementation. The first chapter of this PhD Thesis focuses into tackling the problem of preserving traceability from user requirements to all the models involved in a data warehouse design and back. To this aim, our first objective is to elaborate a trace metamodel for data warehouses that can be used to create and store traceability information.

Recent data warehouse development approaches [25, 9] employ i* [42] in order to elicitate and model user requirements. This approach aims to lessen the gap between IT and business people and allow the data warehouse designer to extract, by means of interviews, the information that decision makers wish to include into the data warehouse.

However, in practice requirements diagrams contain a high number of elements and relationships, including decision makers' goals and information to be stored in the data warehouse. All these elements must be traced to their corresponding multidimensional structures, which can be time consuming task if performed manually.

Moreover, during the reconciliation process, the designer remodels data warehouse structures in order to accommodate as much as possible requirements and data. As decision makers use a different language and mental models than data source designers, it is often the case that concepts use different names and present a different structure.

Thus, in order to maintain traceability in these cases, either all the models must be kept synchronized as changes are performed during the reconciliation process, or, at least, elements must be related in order to be able to trace each requirement to the correct set of multidimensional elements.

Nevertheless, current development approaches either (i) do not refer to these relationships [41, 5], (ii) refer to the existence of these inter-model relationships without explicitly modeling them [9, 38], or (iii) provide implicit traceability by means of name matching [25]. Even in the best case, implicit traceability is only guaranteed until the reconciliation step is performed.

Therefore, by following current development approaches, we cannot ensure that the designer will be able to accurately validate that the implementation of the data warehouse adequately matches user requirements. In order to tackle this problem, our proposal is to explicitly model traceability from user requirements up to the different multidimensional structures that compose the data warehouse. This way, we avoid the aforementioned drawbacks and can automatically evaluate the current status of each requirement.

Traditionally, we can differentiate between two different kinds of traceability. First, we have traceability of a single model as it evolves over time. This kind of traceability records the operations performed over the model in terms of additions, modifications and removals. Second, we have end to end traceability, which traces the relationships between elements in different models. Examples of these relationships are the ones between user requirements and multidimensional structures or when deriving a relational schema from an ER schema. This kind of traceability is ideal for data warehouse development since we are interested in evaluating the suitability of the current data warehouse proposal with respect to user requirements, thus navigating multiple models.

In order to include traceability in the development process, first, we identify the existing relationships between elements in the different models of the data warehouse. Using the approach defined in [25] as an example of hybrid development approach, we identify 4 model to model steps which are depicted in Figure 2.2. In this Figure, user requirements are gathered, modeled [24], and then transformed into an initial multidimensional model (1). This model is then reconciled with the data sources, which are represented by a logical model composed by tables (2). The result is a hybrid multidimensional model which includes information from both user requirements and data sources (3). Finally, undesired elements are filtered or modified and the final multidimensional model for the data warehouse is obtained (4).

In order to categorize the different traces created in this process, we reviewed the literature on traceability in both Requirements Engineering (RE) and Model Driven Development (MDD) [36, 1, 2, 12, 11, 31, 43, 1, 16, 29, 39, 40]. The result of this review was a classification of traces according to eight different categories defined in RE [36]. This classification was aligned with the data warehouse development process in Chapter 3 and further specialized in Chapter 4. The initial classification is shown in Figure 2.3 and is comprised by six basic semantic types:

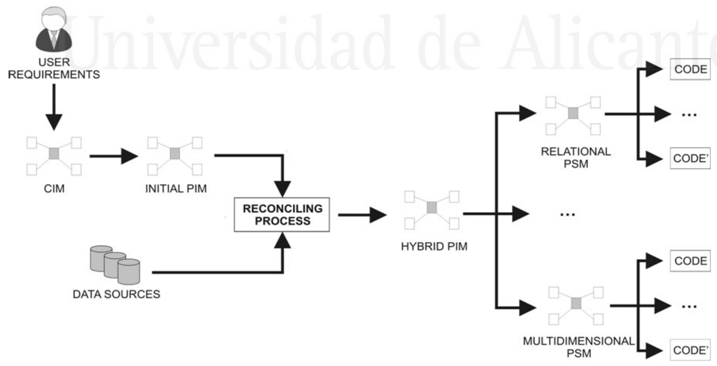


Figure 2.2: An example of hybrid development approach from [25].

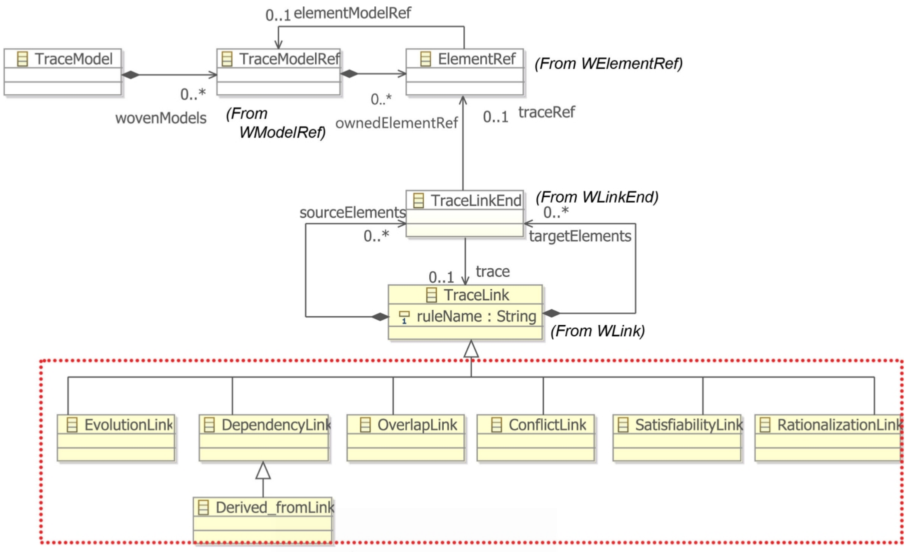


Figure 2.3: Traceability metamodel for explicit model of traceability in data warehouses [22]

1. **Satisfiability** captures the relationship between requirements and multidimensional elements. Their source must always be a requirement that has one or more multidimensional counterparts that satisfy it. These multidimensional counterparts are always the targets of the link, as their existence is tied to the requirements of the user. In our approach, the only element of the diagram that does not have a counterpart are the user goals. As their satisfaction depends on the existence of the necessary information to take decisions, they can be evaluated by checking if the elements traced to the data warehouse are implemented.

2. **Derived_from** captures the relationship between data sources and multidimensional elements. Their source must always be one or more data source elements, such as tables, columns, key, etc. represented at the logical level, and their targets are one or more multidimensional counterparts obtained by reverse engineering the logical model.

3. **Evolution** captures the links between elements in one multidimensional model and their corresponding version in the next model. These links are designed specially for approaches such as [25] where there are multiple multidimensional models involved in the development process. For example, the existence (or absence) of an evolution link from one element

the reconciled model to one in the final model determines if the element was chosen for inclusion in the data warehouse or was discarded from the implementation.

4. **Overlap** captures matches between the multidimensional elements expected by user requirements and those obtained by means of reverse engineering from data sources. These will be analyzed in detail in the next section.

5. **Conflict** captures mismatches between the multidimensional elements expected by user requirements and those obtained by means of reverse engineering from data sources. These will be analyzed in detail in the next section.

6. **Rationalization** captures relationships that are not covered in a canonic data warehouse development. For example, they relate engineered solutions to conflicts with their original elements, as well as other additions performed by the designer.

Once we have defined the set of trace categories, we need to define a traceability metamodel or adapt an existing one that is (i) loosely coupled with data warehouse models, since no standard for user requirements nor multidimensional modeling has been defined until now, thus it would limit the applicability of the approach, and (ii) avoids polluting the models with traceability information, as it can be misleading for the data warehouse designer

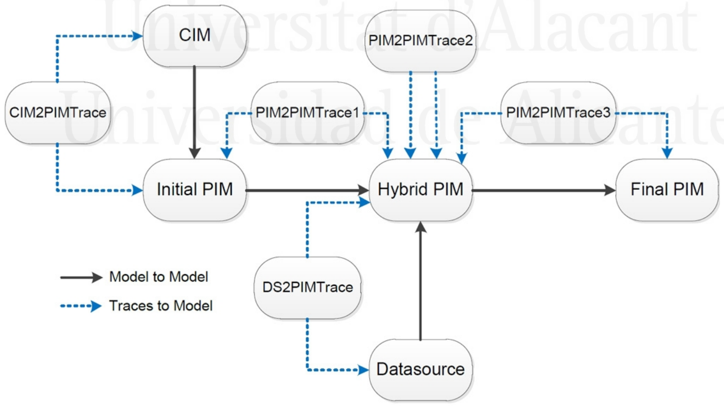


Figure 2.4: End-to-End traceability models included into the hybrid development process.

[16]. After performing a review of the existing proposals, we obtained the traceability metamodel shown in shown in Figure 2.3, which specializes the general proposal from [6].

The traceability model shown is highly flexible, as it allows the definition many-to-many traces, involving as many different models as required. Additionally, the model is loosely coupled, since it works by using element and model references, thus avoiding the insertion of traceability information into the data warehouse models. Therefore, it can be applied to enhance any of the current data warehouse development approaches.

However, traceability can introduce a considerable overhead in any process if it has to be captured and maintained manually. Thus, in order to avoid introducing overhead in an already time-consuming development process, we seamlessly integrate the generation of traces with a MDD based approach. This way, we automatically generate traces at the same time that we derive the model in the next step of the process, as shown in more detail in Chapter 3. By introducing traceability in the process, we obtain a refined development process that includes the set of trace models on top of the traditional ones for data warehouse development. The whole set of models is presented in Figure 2.4. In this Figure, we can see the data warehouse and the different traceability models from a MDD perspective. Requirements are gathered at the CIM (Computation Independent Model) layer. Then, they are derived into

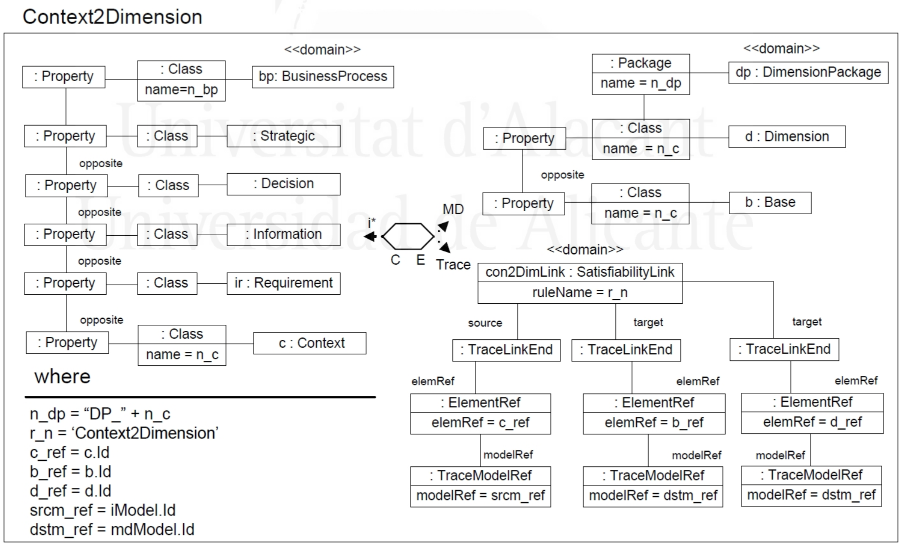


Figure 2.5: A QVT rule enhanced with traceability information.

the multidimensional models at the PIM (Platform Independent Model) layer. Finally, these models are reconciled with information from the data sources, that is represented at the PSM (Platform Specific Model) layer and the final multidimensional model is obtained.

The new traceability models introduced link user requirements and data sources with multidimensional models. This allows us to trace multidimensional and data source elements to user requirements and identify (i) which user requirements cannot be satisfied, and (ii) which elements are affected by a change and propagate the change if the designer wishes to do so. These models are automatically generated by means of Query/View/Transformation (QVT) rules [28]. QVT is a standard proposed by the Object Management Group for model to model transformations. A QVT rule, such as the one in Figure 2.5 checks for the existence of the pattern in the left hand side of the rule. If this pattern is found, then the right hand side of the rule is executed, creating the corresponding elements on each model. The implementation of these QVT rules in our CASE tool is done by means of the ATLAS Transformation Language (ATL) [17].

Conceptually, user requirements are linked to multidimensional elements, as shown in Figure 2.6. These links are kept up to date thanks to a reactive framework that observes and records changes performed over the models and then allows the possibility to propagate them, including the automatic update of trace models. In practice, the trace models can be inspected by the data warehouse designer, as well as used as input in other model to model transformations that require traceability information. However, for data warehouse design, they are intended to be transparent for the user, and allow us to perform different tasks such as change propagation or impact change analysis, as shown in Chapter 3.

2.4.2 Traceability during the Reconciliation Process

In order to trace user requirements up to the data warehouse implementation, requirements have to be traced through the reconciliation process. During this step, some multidimensional structures may be changed or even discarded by the designer due to the differences between the expected structure of the data and the real one. Therefore, the reconciliation step requires special attention, since it is often left up to the expertise of the data warehouse designer and not performed systematically nor detailed enough to allow us to analyze and validate the reasoning behind the process.

The reconciliation step takes as input a multidimensional model specified by user requirements and a set of data sources to supply data for that multidimensional model. The objectives of this step are (i) verify that the data required to populate the structures in the multidimensional model are available, and (ii) find additional data which may be relevant for taking decisions

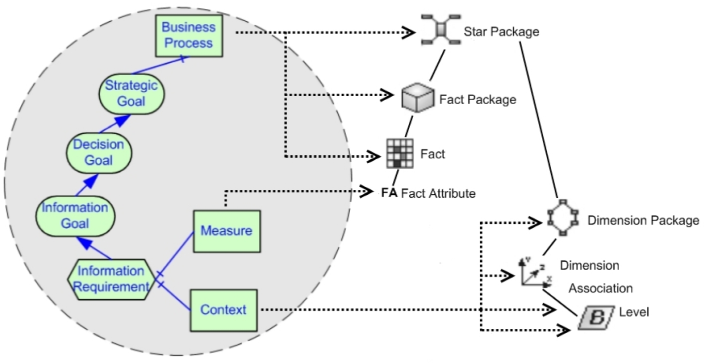


Figure 2.6: Conceptual trace links between user requirements and multidimensional models.

but has been overlooked during the requirements elicitation step. During this step, elements may either be discarded, as there is no data available, confirmed, added, because they may be relevant for the decision making process, or modified because the data cannot be transformed according to what user requirements expect. Current data warehouse development approaches check the availability of the information by name matching [25], by applying multidimensional normal forms [26], by modeling the business activity [9], or by employing ontologies [32]. However, any mismatch between the data warehouse schema and the data sources that does not follow these patterns must be solved by the data warehouse designer.

Current data warehouse approaches do not explicitly consider these mismatches and provide no tools to model them. Thus, the designer has to perform all the operations according to his own experience, the desired multidimensional schema, and the data sources available. Unfortunately, these mismatches happen often for several reasons:

1. First, naming conventions used by decision makers to name concepts rarely match names in operational data sources.

2. Second, the structure of concepts and attributes in the data warehouse schema is unlikely to match with the data sources, even when reverse engineered to a multidimensional view. For example, consider a database with a table for storing the detail of orders and has a foreign key to another table that contains the list of orders. Such structure is typical in many transactional databases, and can be seen, for example, in the sample database Northwind Traders (Figure 2.7). Applying multidimensional normal forms to these kind of sources results in an incorrect mul-

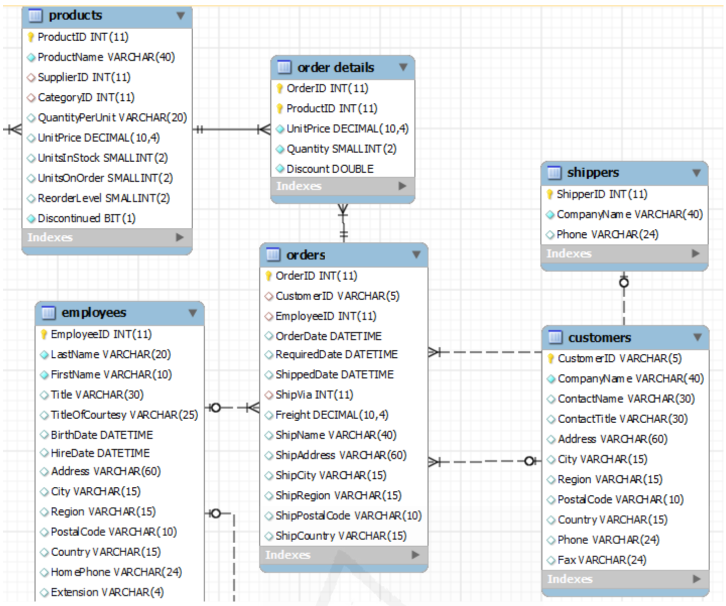


Figure 2.7: Excerpt of the transactional database Northwind Traders.

tidimensional schema, identifying orders as a dimension and customer and supplier as hierarchy levels of such dimension.

3. Third, not all the information is any longer located exclusively in operational sources, thus modeling business processes has a limited scope. For example, a current challenge in enterprises is breaking information silos, each presenting a different structure to store the data.

As a result of all these factors and the lack of tools to document the process, the reconciliation step behaves like a black box. Therefore, it presents several shortcomings such as loss of traceability, difficulty to understand the information provided by data sources, and difficulty to incorporate new data sources among others.

In order to avoid these drawbacks, we propose a process to preserve traceability during the reconciliation step. Our proposal is to keep both the required and the reverse engineered elements separate. Then, instead of having the designer merge them through combinations of add, delete and modify operations, we model the end-to-end flow of information, that will be later implemented in Extraction/Transformation/Load (ETL) processes. The key differences between our approach and ETL processes is that the links are (i) established at conceptual level using a multidimensional view, (ii) created at design-time, whereas ETL process are created after the implementation of the data warehouse, and (iii) they establish end-to-end connections but do not specify how information is transformed or filtered. This way, the designer can model the relationships between the expected and reverse engineered elements, and we can provide additional analysis what kind of mismatches are present in the data warehouse structure. To this aim, first, we will focus on defining the basic relationships between elements that affect the derivation of the reconciled data warehouse model. Then, in the following section, we will analyze these relationships in-depth. The possible basic relationships are defined as follows:

1. One or more data source elements may provide information for one or more data warehouse elements without requiring modifications. In this case, the relationship established between them is defined as an *Overlap*. As a result, user requirements satisfied by the multidimensional elements are fully supported by the data warehouse implementation.

2. One or more data source elements may provide information for one or more data warehouse elements requiring modifications. In this case, the relationship established between them is defined as a *Conflict*. An example of such modification is when data is aggregated in the data warehouse with respect to data sources. Another example is when data is more aggregated than required by the data warehouse, and the resolution of the data warehouse schema is reduced in order to provide at least some degree of information.

3. One or more conflicts may be solved by creating a reconciled element. In this case, the relationship established between elements is defined as a *Rationalization* performed by the designer.

After defining the basic relationships, we can relate the flows of information between the expected schema and the reverse engineered one. Thus, the next step in the process is to analyze each element required and relate it to the corresponding (if any) data source elements that provide the necessary information. The result of this step can be seen exemplified in Figure 2.8. In this Figure, the expected data warehouse schema of a university is related to the reverse engineered one. As we can see, both dimensions are marked as in conflict due to a mismatch in the way of identifying their hierarchy levels. Furthermore, we can also see that the attribute "Name" from teacher is divided into three different attributes in the data sources and, thus, requires a transformation. Although the names and structure of both schemata do not fully match, we obtain a clear view of the elements involved in the reconciliation process. In the following section we will provide more details on the rationale of the relationships and formalize their meaning.

Once all elements have been related, we review the existing conflicts and either (i) chose one element as a solution, or (ii) create a reconciled element

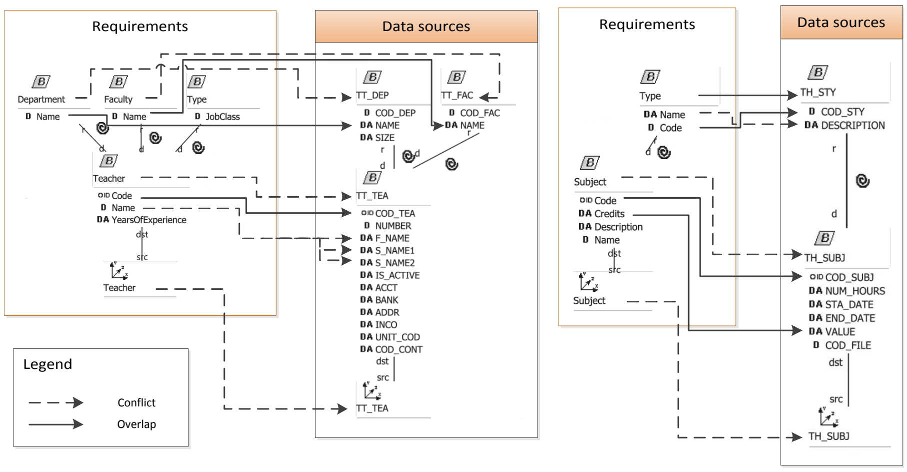


Figure 2.8: Example of mappings between the expected concepts and the reverse engineered ones

and relate it to the conflicting elements by means of a *Rationalization* link. Finally, we select the elements we wish to include into the data warehouse implementation. In order to avoid introducing overhead in the process, the creation of the final multidimensional schema can be automated by filtering non-selected elements and following two rules in the derivation process:

1. If two or more elements are related by an *Overlap*, if any of them is selected as a candidate for the final data warehouse schema, then a corresponding element is created and both requirements and data source elements are traced to the final schema by means of *Evolution* traces.

2. If two or more elements are related by a *Conflict*, if any of them is selected as a candidate for the final data warehouse schema, then a corresponding element is created and only the selected elements are traced into the final schema by means of *Evolution* traces.

These two rules can be automated, as it is shown in Chapter 4, by coding their logic into QVT rules [28], thus fully automating the derivation process. Moreover, by following these two rules, we can identify whether a user requirement is fully supported, partially supported, or discarded in the final data warehouse design. Fully supported requirements are those that are traced to overlapping or conflicting elements that were selected and derived in the final data warehouse. In the first case, the corresponding data source element was compliant and thus the data is available. In the second case, although the corresponding data source is not compliant, selecting the element corresponding to requirements means that its structure will be respected, and thus data will be transformed. Additionally, if the designer provides a reconciled element for a conflict, we also consider that the user requirement has been fully supported, as a solution was provided. On the other hand, partially supported requirements are those that are traced to conflicting elements whose conflicting counterparts were selected. Thus, the data available acts as a substitute of the expected data. Finally, unsupported or discarded requirements are those that cannot be traced to the final data warehouse schema through any element.

After defining the process to preserve traceability through the reconciliation step and derive the final data warehouse schema, we will proceed to analyze in-depth the relationships between expected multidimensional structures and reverse engineered ones, and formalize them. This will serve us for various purposes: (i) to gain a better understanding of the status of user requirements in the development process, (ii) to be able to semi-automate the creation of trace links during the reconciliation process, and (iii) to provide the data warehouse designer with rich semantics to adequately document the reconciliation process and serve as a basis for future ETL processes.

Modeling Data Integration during the Reconciliation step

The relationships defined in the previous section are oriented to preserve traceability. However, they are not detailed enough to provide the designer tools to adequately document the reconciliation step. Current data warehouse development approaches rely on Extraction/Transformation/Load (ETL) processes as a means of documenting the results of the reconciliation step. However, this approach has several shortcomings. First, ETL processes connect data sources to the result of the reconciliation step, once the data warehouse has already been implemented, instead of with the data warehouse schema specified by user requirements. Therefore, they cannot be used to evaluate if a new data source provides new, previously unavailable information that enables unsatisfied user requirements. Additionally, this also means that they cannot document any mismatches between what was expected and what is stored in the data warehouse. Second, ETL processes specify the flow of information from data sources to tables in the target database. Therefore, they ignore the multidimensional characteristics of data. Third, ETL processes do not provide a global overview of the relationships. Instead, they are divided into several files with different refreshing intervals. In turn, the reconciliation step is poorly documented.

The lack of documentation can become an issue when incorporating new data sources into an existing data warehouse. If the existing documentation is insufficient, a manual inspection of already existing data sources may be required, which is an extremely time consuming and error-prone task. Unfortunately, this task is becoming more common nowadays. On the one hand,

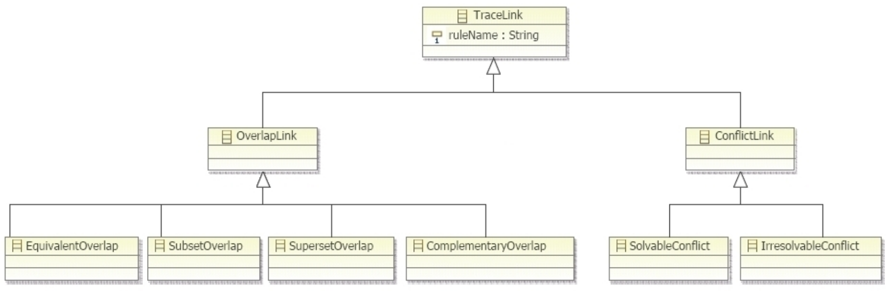


Figure 2.9: Classification of relationships between user concepts and data sources

the Big Data trend is gaining strength, which implies the addition of information from new external sources, such as social networks or RDF resources. On the other hand, many companies already have a data warehouse. When companies fuse together with other companies, they require to integrate all the information available into a single data warehouse, thus requiring accurate information about the data stored. In practice, what happens is that companies are unable to perform this task, and information silos start appearing, making it difficult to share information and compare data throughout the enterprise.

In order to tackle this problem, we specialize the relationships defined to preserve traceability, allowing the data warehouse designer to specify the concrete semantics of the relationships between the data warehouse and the data sources. In order to cover all the possibilities and provide a complete set of categories, we model the relationships from the Set Theory point of view. In our approach, we refer to multidimensional elements specified by requirements as expected elements or user concepts. For each user concept, the designer specifies a domain D from which the concept can take values. Then, this set of values is compared with the domain defined by the values stored in the data sources. The result of this comparison is a single category from our classification. The complete classification is shown in Figure 2.9, specializing the *Overlap* and *Conflict* base categories. As a baseline, *Overlap* is defined as a match between source and target domains. Conversely, a *Conflict* implies a mismatch between source and target domains. Furthermore, in the following, we will refer to elements in the expected data warehouse as trace sources and to the elements from the data sources as trace targets[1].

1. An *Equal Overlap* occurs when both source and target domains share

[1]In practice, it is easier to systematically check each element in the expected data warehouse schema than to check each element in the data sources

the same elements.

2. A *Subset Overlap* occurs when the target domains are missing some elements from the source domains. For example, if we expect the identifier of a document, “idDocument” to contain identifiers from various digital libraries but the actual data contains only identifiers from one library.

3. A *Superset Overlap* is the reverse relationship of a *Subset Overlap*.

4. A *Complementary Overlap* occurs when source and target domains include some elements which do not appear on the other domain.

5. A *Solvable Conflict* occurs when source and target domains do not match but exists a function F which can project elements from the target domains into source domains. For example, an attribute “language_code” does not share the same domain as “language”, but can be translated with the aid of a language code table.

6. An *Irresolvable Conflict* occurs when no function F exists or it is currently unknown.

Each of these categories can be applied to different levels of detail in multidimensional schemata, from attributes to dimensions. The specific formalization of each category and abstraction level can be found in Chapter 4. By using this set of categories, the designer can choose to specify the exact nature of the relationship between the data warehouse and the data sources during the reconciliation step. Furthermore, the designer has to relate only the attribute level, as the rest of the abstraction levels are calculated automatically.

We have successfully applied our approach to a case study of a digital library at the University of Alicante. In our case study, we wished to build an analysis model of the documents stored at the digital library. However, due to the evolution of standards and ontologies used in digital libraries, there was information scattered through several attributes, hierarchy levels and dimensions. Thanks to our approach, we could obtain a clear view of the information that was necessary to gather for each required dimension, if a transformation was necessary or not, and what information was missing.

For example, as shown in Figure 2.10, we required information from two different dimensions in order to obtain all the necessary data for the “Document” dimension. One of this dimensions contained a complementary level to the first one we required, as there were some attributes missing and other unexpected attributes included, whereas the other dimension contained an equivalent level to the second one required. For a complete description of the case study please see Chapter 4.

Compared to current approaches, our proposal has several benefits. First, all the relationships are gathered into a single model, which provides an

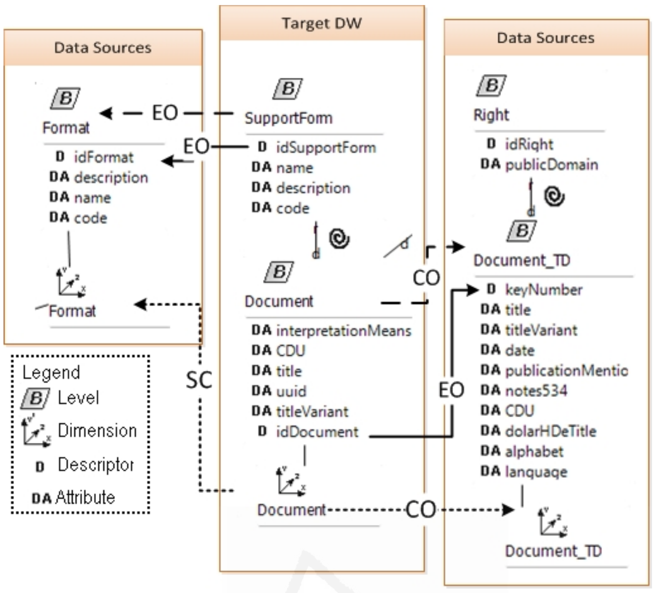


Figure 2.10: Example of detailed traceability between the expected schema and the reverse engineered from data sources.

overview of all the elements involved in the process. This makes it easier to plan how to integrate new data sources into the process. Second, we can analyze more accurately how a change in the data sources will impact the data warehouse. For example, modifying the values loaded into a level identifier may greatly affect the data warehouse, as it alters the result of the aggregation. Third, it allows the designer to highlight important information, such as if any elements are missing certain information. For example, in some cases, decision makers have reported anomalies while analyzing values of Key Performance Indicators (KPIs). Many times, these anomalies are due to a partial lack of data in the operational data sources and, thus, the real value that decision makers expect is not loaded into the data warehouse.

Thanks to the combination of our formalization for the reconciliation step and the traceability metamodel, we are able to accurately trace and evaluate the impact of a change in any user requirements. This includes changes performed not only in the data warehouse, but also those performed in the data source. Therefore, our enhanced methodology for data warehouse development can support better the constant evolution of data warehouses and rely less on the data warehouse designer experience.

2.4.3 Testing and Improving User Requirements Diagrams

In the previous sections we have tackled in detail the problem of traceability loss during data warehouse development. Thus we have covered initial objectives of (2) Lowering the complexity of managing several models involved in data warehouse development, (3) Ensuring that the resulting data warehouse models adequately fulfill the specified user requirements, and (4) Supporting the evolution of the data warehouse once it has been implemented. These points are improved thanks to the navigation capabilities provided by the trace links, to the reactive framework that allows to propagate updates, and to the modified reconciliation process that allows the designer to have a better understanding of the data involved in the process. Now, in the following two sections we will focus on (1) Improving requirements validation.

Despite the addition of goal-based requirements models in recent approaches, such as [25] and [9], the requirements elicitation step still presents problems. First of all, decision makers are not technical people, and while goal models are understandable by them, they tend to think in terms of measures and Key Performance Indicators, rather than in terms of goals. Thus it is important to keep the models simple or, at least, focused, in order to adequately elicitate requirements. Unfortunately, recent approaches make use of i* as the basic framework for gathering requirements, which lacks any kind of modularity aside from the Actor/Role elements. In turn, the size of the models becomes a problem for for correction and communication with the users. In practice, the size of the diagram increases to the point that even the data warehouse designer misses some of the elements already defined and duplicates them with different structure. Therefore, it is important to improve this aspect in order to manage corrections and changes in data warehouse requirements in an easier way.

We propose to address this problem by improving the modularity of i* in the data warehouse field, and allowing the designer to partition the diagram whenever necessary, while at the same time maintaining the semantics of each partition. To this aim, we extend the i* for data warehouses requirements model [24] and define a set of modules that support this goal. First, we present the elements involved in requirements elicitation by means of an example, shown in Figure 2.11. Then, we describe the basic process followed in order to elicitate requirements and present our modules extension.

In our example, we start the requirements analysis from a business process (BP), related to the decision maker. The BP, which is the center of the analysis, models an activity of interest for the decision-maker. In this case the activity is to *Make Contracts*, and has associated a series of strategic goals, aimed to improve the business performance. Strategic goals represent the highest level of abstraction. They are thought as changes from a current situation into a better one in terms of business process objectives. In our case, the strategic

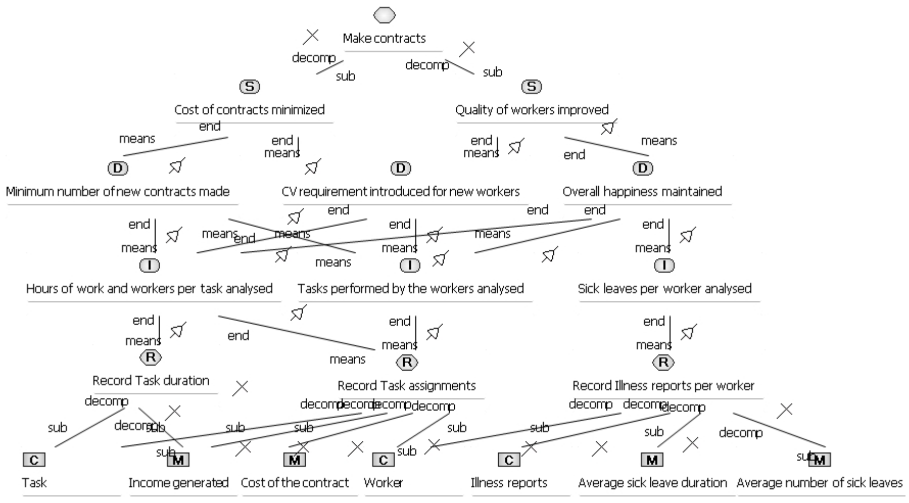

Make contracts
decomp
sub
S
Cost of contracts minimized
Quality of workers improved
means
end
D
Minimum number of new contracts made
CV requirement introduced for new workers
Overall happiness maintained
I
Hours of work and workers per task analysed
Tasks performed by the workers analysed
Sick leaves per worker analysed
R
Record Task duration
Record Task assignments
Record Illness reports per worker
C
M
Task
Income generated
Cost of the contract
Worker
Illness reports
Average sick leave duration
Average number of sick leaves

Figure 2.11: Example of i* model for data warehouses.

goals associated with the BP are *Cost of contracts minimized* and *Quality of workers increased*. Other examples of strategic goals would be *Increase sales*, *Increase number of customers*, *Decrease cost*, etc. Their fulfillment causes an immediate benefit for the organization.

In order to achieve these strategic goals, there are a series of decision goals that must be met. Decision goals represent the medium level of abstraction in our SR (Strategic Rationale) models. They try to answer the question "how can a strategic goal be achieved?", and they aim to take the appropriate actions to fulfill a strategic goal. They are related to strategic goals by intentional means-end relationships. In our example, in order to achieve *Cost of contracts minimized*, it has been decided that it is necessary to have the *Minimum number of new contracts made* as well as have a *CV requirement introduced for new workers*, in order to achieve the strategic goal. However, decision goals can affect more than one strategic goal. In our case, the last decision goal is related with *Quality of workers increased* strategic goal as well, since the CV affects the quality of the new workers being employed. Other examples of decision goals would be *Determine some kind of promotion* or *Open new stores*. Their fulfillment only causes a benefit for the organization if it helps to reach strategic goals, since decision goals only take place within the context of strategic goals.

As with the strategic goals, the decision goals can be achieved by having the necessary information available. This required information is modeled by means of the informational goals. Information goals represent the lowest level of abstraction. They try to answer the question: "how can decision goals be achieved in terms of information required?", and they are related to the information required by a decision goal to be achieved. In our example, the information required is *Hours of work and workers per task analysed* and *Tasks performed by the workers analysed* for each decision goal, whereas the information about *Sick leaves per worker analysed* affects only the *Overall happiness maintained* decision goal. Other examples of information goals are *Analyze customer purchases* or *Examine stocks.* Their fulfillment helps to achieve decision goals and they only happen within the context of decision goals.

Finally, informational goals are achieved by means of information requirements. In our case, we need to *Record Task duration*, *Record Task assignments*, and *Record Illness reports per worker* in order to gather the required information. Each of these requirements is decomposed into contexts and measures, that represent the information to be stored in the data warehouse. The example includes the *Task, Worker*, and *Illness report* contexts as well as the *Income generated, Average sick leave duration* and *Average number of sick leaves* measures, which determine the performance of the business process.

As we have seen, requirements models go from the highest level of abstraction towards lower levels of abstraction. The basic steps of the requirements elicitation process are as follows: first, the process starts with the identification of a target business process that the decision maker wishes to improve. Then, the decision maker describes the goals that she wants to fulfill in order to improve the business process. These strategic goals are then refined into decision goals and information goals that represent decisions to be taken and information to be obtained. This refinement is achieved by asking the questions "how¿' and "why?" to help in the exploration. Finally, the necessary information requirements that support information goals are identified. These information requirements gather the different entities required to achieve an information goal.

As we can observe, lower level abstraction goals only appear in the context of one or several higher abstraction goals. Furthermore, we can observe a differentiation between the rationale of the goal tree and the information entities that will support these goals. With these considerations, our modularization proposal is aimed to (i) allow a semantic partition of user requirements diagrams and (ii) separate the business concerns (goals) from the entities that will be captured in the data warehouse. This way, the designer can focus on refining the goal models looking for additional goals or improving the detail of the hierarchies and measures to be included in the data warehouse.

The modules defined can be seen in Figure 2.12, where they extend from the *Package* and include an abstract class *iModule* in order to help with the definition of OCL constraints that guarantee their correct application.

- **Decision modules** include the elements related to a given *decision goal.*

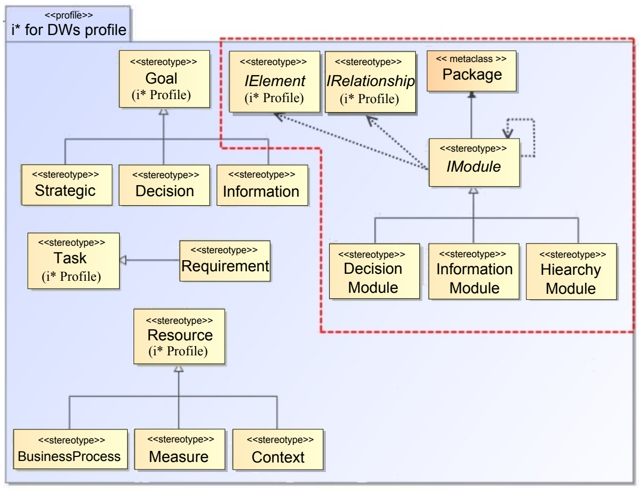

<<profile>>
i* for DWs profile
<<stereotype>> Goal (i* Profile)
<<stereotype>> IElement (i* Profile)
<<stereotype>> IRelationship (i* Profile)
<< metaclass >> Package
<<stereotype>> IModule
<<stereotype>> Strategic
<<stereotype>> Decision
<<stereotype>> Information
<<stereotype>> Task (i* Profile)
<<stereotype>> Requirement
<<stereotype>> Decision Module
<<stereotype>> Information Module
<<stereotype>> Hiearchy Module
<<stereotype>> Resource (i* Profile)
<<stereotype>> BusinessProcess
<<stereotype>> Measure
<<stereotype>> Context

Figure 2.12: i* profile with modules extension for DW

They can include *decision goals, information goals, requirements, contexts, measures, other decision modules, information modules,* and *hierarchy modules.* They contain all the necessary information to take a given decision, which helps achieving a *strategic goal.*

- **Information modules** include the elements related to a given information goal. They can include *information goals, requirements, contexts, measures, other information modules,* and *hierarchy modules.* They aggregate all the information which is necessary to satisfy a given information goal. *Note: This module was redefined into Information Requirements module after a deep analysis of the experiment results.*

- **Hierarchy modules** include the elements which constitute a hierarchy. They are formed by the different contexts which represent the different levels of aggregation of a dimension. They can only include *contexts.* These modules help with the reusability of the dimensions at the requirements level, and hide the complexity of hierarchies when it is unnecessary.

In addition to the definition of these modules, we provide a set of recommendations to aid in their application. The complete list of recommendations can be found in Chapter 5.

After defining the set of modules, we performed an experiment in order to evaluate their suitability of the new constructs. This experiment was performed by comparing the effectiveness and perceived quality of the models with and without modules when performing several tasks related to modeling and identifying user requirements. The complete description and results of the experiment can be seen in Chapter 5. The analysis of the results shown that (i) the rate of errors dropped when identifying elements in moderate sized diagrams, and (ii) participants using the modules were systematically missing measures when asked to identify relevant concepts for the underlying data warehouse in the diagram. Taking into account these results, we lowered the abstraction level of *Information modules* and substituted them by *Information Requirements modules*, thus effectively separating decision maker goals from all the entities captured in the data warehouse.

Thanks to the definition of semantic modules we have obtained an improved metamodel for user requirements that allows designers to manage diagrams in an easier way and communicate better with decision makers. Finally, in the next section, we will tackle the problem of further lessening the gap between data warehouse designers (IT) and decision makers, dealing with the lack of a Business Intelligence strategy and of how to manage the different partial views provided by individual decision makers.

2.4.4 Aligning the Data Warehouse with the Corporate Strategy

Even the best designed data warehouse may fail to support and improve business performance if (i) decision makers are not able to understand the meaning of data and translate the information to their business goals, or (ii) the data warehouse is not correctly aligned with the business strategy. This statement is supported by diverse studies and surveys, such as [4], where the Gartner Group highlights that one of the main reasons for the high rate of failure of data warehouses is the language gap between IT and business people.

Recent data warehouse development approaches [25, 9] go as far as elicitating user requirements in terms of decision makers' goals. However, previous studies have pointed out that it is not feasible to extract a comprehensive and accurate set of requirements from decision makers [41], since each decision maker can only provide a partial, personal, point of view. Thus, in order to tackle the aforementioned problems, in this PhD Thesis we propose to enrich current data warehouse development techniques by including the corporate strategy into data warehouse development.

The last challenge faced by current data warehouse development approaches is validating user requirements and ensuring that the data warehouse will adequately support the business plan. Current development approaches [25, 9] elicitate requirements from individual decision makers but pay little attention to business goals. In the best case, a partial modeling of the current business activity is performed [9] in order to aid with the reconciliation process. However, business objectives, strategies, and plans are completely overlooked.

Including the business plan into the development process is a challenging task. Most of the information available is written in informal language and, thus, can not be used directly. Therefore, the first step is to formalize the knowledge included in the business plan. In order to tackle this problem, in this PhD Thesis we propose to support this step by using the Business Motivation Model (BMM) [27]. BMM is a standard proposed by the Object Management Group that captures the basic semantics of the elements involved in a business plan. These elements range from goals to initiatives, including the Mission and the Vision of the organization. However, these elements are only partially formalized in BMM. Each element includes only its type, an identifier and a textual description. Even the relationships between elements are optional. Therefore, staying at this abstraction level limits the information that can be extracted from these elements. For example, given the goal "To be a Premium Brand car rental company" and the objective "Be rated higher than 6 by AC Nielson in top car rental companies in EU" there is no way evaluate the time to target left for the goal nor if it is satisfied or not according to its objective, since we have no *current value*, *target value* or *formula* attributes that allows us to calculate this information.

Nevertheless, this pitfall can be avoided as shown in Chapter 6 by further formalizing the information captured in BMM. A BMM can serve as an intermediate representation to extract the initial structure of the plan. Then, it can be further formalized, for example, by using the Business Intelligence Model (BIM) [3]. The Business Intelligence Model is a metamodel designed to represent business goals and strategies. Unlike i* models used in data warehouses, it does not focus on individual decision maker goals, but rather on the objectives of the enterprise as a whole, and provides fully defined constructs for each element included in the model.

We will introduce the basic concepts by briefly describing an example of a BIM model shown in Figure 2.13. In this example, the company EU-Rent wishes to achieve several high level, long-term *Strategic Goals*, denoted as (SG). EU-Rent wishes to be positioned as a "Premium Brand Car rental company" (SG1) and provide "Industry-leading customer service" (SG2).

Goals, specially strategic ones can have *Indicators* that serve to monitor their performance. In this case, we can see that EU-Rent has two different indicators for SG1. The first one, more specific, is "Be rated higher than 6 by AC Nielson in top car rental companies in EU" (xg1.1). The second, more general, is "Be rated higher than 9 by AC Nielson in top car rental companies" (xg1.2). These indicators are used by the company to analyze its current performance and alert about potential deviations from the plan.

Strategic goals are supported by middle level abstraction, middle-term *Operational Goals*. In order to achieve "Be Premium Brand car rental company" (SG1), it is positive if the enterprise "Operate nation-wide in each country" (OG1). These operational goals focus on aiding in the satisfaction of strategic goals and are further decomposed and supported by *Tactical Goals*.

A tactical goal represents a short-term goal that can be directly operationalized by the business processes of the enterprise. For example, "Encourage rental extension" (TG1) can be part of "Operate nation-wide in each country" (OG1) and operationalized by the "Car Rental" process. While the BIM metamodel also supports the addition of business processes, not all business plans include this level of detail, as in the case of the EU-Rent scenario.

Finally, goals can be affected by different *Situations* that may help or hurt them. A situation represents a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis [13] of the external and internal factors that may affect the accomplishment of a goal. For example, the presence of "Budget Airlines" (S4) may hurt the goal "Operate nation-wide in each country" (OG1), as budget airlines are a strong competitor to renting a car for long distances within a country.

By means of these constructs, we can further formalize the business plan, obtaining as a result a fully formalized business strategy. Once we have formalized the information stored in the business plan, we can include this information into the data warehouse development process in order to validate user requirements.

In order to perform this task, we follow the approach described in Chapter 6, consisting in following steps. First, we perform an initial requirements analysis step by elicitating requirements from the different decision makers. In our case, we perform this step by using our improved i* profile for data warehouses [23]. Afterwards, we obtain a business strategy model from the business plan that includes the business goals. Once we have gathered both user requirements and the business strategy model, we align the different decision makers'

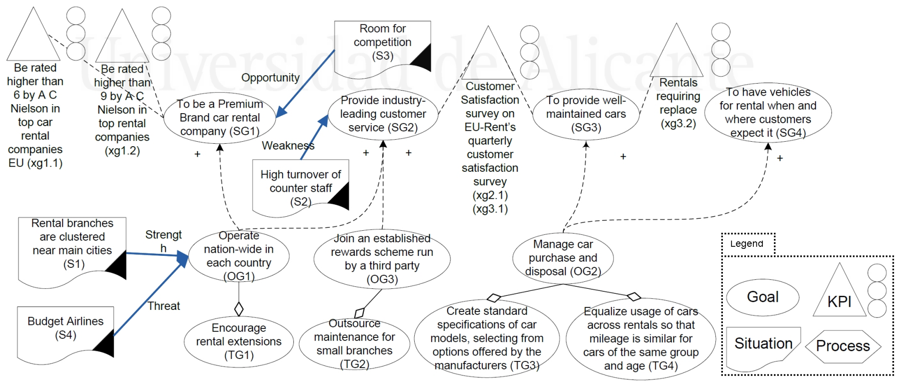

Be rated higher than 6 by A C Nielson in top car rental companies EU (xg1.1)
Be rated higher than 9 by A C Nielson in top rental companies (xg1.2)
Opportunity
Room for competition (S3)
To be a Premium Brand car rental company (SG1)
Provide industry-leading customer service (SG2)
Weakness
High turnover of counter staff (S2)
Customer Satisfaction survey on EU-Rent's quarterly customer satisfaction survey (xg2.1) (xg3.1)
To provide well-maintained cars (SG3)
Rentals requiring replace (xg3.2)
To have vehicles for rental when and where customers expect it (SG4)
Rental branches are clustered near main cities (S1)
Strength
Operate nation-wide in each country (OG1)
Join an established rewards scheme run by a third party (OG3)
Manage car purchase and disposal (OG2)
Threat
Budget Airlines (S4)
Encourage rental extensions (TG1)
Outsource maintenance for small branches (TG2)
Create standard specifications of car models, selecting from options offered by the manufacturers (TG3)
Equalize usage of cars across rentals so that mileage is similar for cars of the same group and age (TG4)
+
Legend
Goal
KPI
Situation
Process

Figure 2.13: Example business strategy modeled after the EU-Rent scenario.

goals with the business strategy model, as exemplified in Figure 2.14. In order to perform this alignment it is necessary to (i) apply a series of restrictions described in Chapter 6, and (ii) involve a domain expert in order to guarantee a correct alignment. Finally, we proceed to analyze the results.

As a result of the alignment, we can identify which business goals are being considered by the decision makers. For example, in Figure 2.14, we can see how the "Customer Relationship Manager" is focusing on the goal "Provide industry-leading customer service" (SG2) and the "Vehicle Manager" is actively considering the "Provide well-maintained cars" business goal (SG3). Thus, we say that SG2 and SG3 are currently being *Supported* by the data warehouse and by the decision making process. Additionally, we can also identify that no decision maker has currently any goals aligned with "To be a Premium Brand car rental company" (SG1) and "To have vehicles for rental when and where customers expect it" (SG4). Thus, we say that SG1 and SG4 are *Unsupported* by the data warehouse and by the decision making process. Thus, the data warehouse may be missing information required to make decisions regarding these goals that should be included. Furthermore, we can identify that there are no two decision makers working towards the same business goal, thus there is, initially, no information to be shared or potential collaboration. Finally, we can see that there are no unaligned user goals, thus the user requirements can be considered valid from a business point of view and no correction is needed.

2.5 Discussion and Conclusions

In this PhD Thesis we have analyzed the current pitfalls in data warehouse development and have presented the building blocks for an enhanced data warehouse development approach that avoids these pitfalls. The set of building blocks includes a traceability metamodel that can support traceability through the whole development process, a formalization to allow the designer to document and derive a reconciled data warehouse, a modularization proposal for improving communication with users using i* based goal models, and an alignment method to validate user requirements and ensure that the data warehouse is aligned with the business plan. Furthermore, we have integrated our techniques in an enhanced data warehouse development approach using [25] as baseline. Additionally, our proposal can be combined with most of the existing data warehouse development approaches, thus allowing flexibility in the choice of the set of models used.

The results obtained in this PhD Thesis include:

- Preservation of traceability in data warehouse development.
- Formalization and documentation of the reconciliation process.

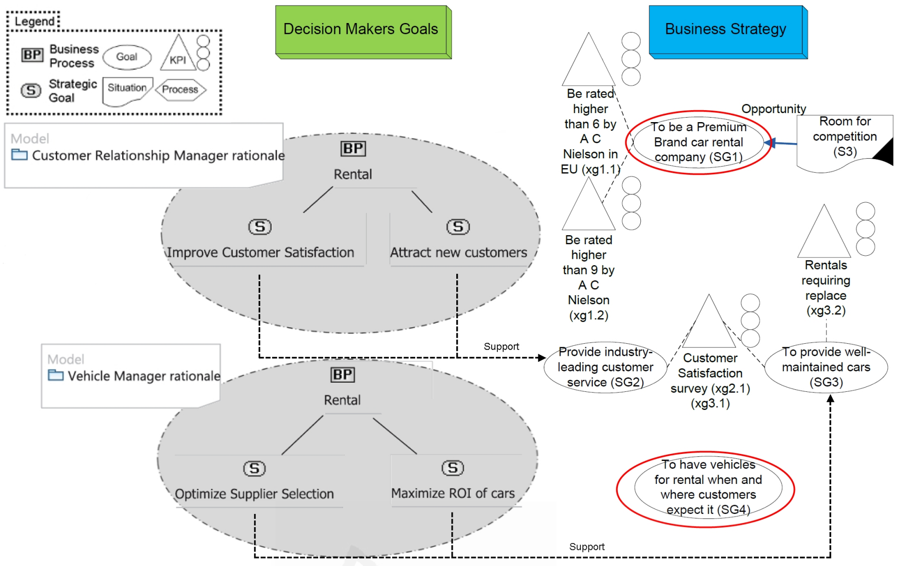

Legend
BP Business Process
S Strategic Goal
Goal
KPI
Situation
Process
Decision Makers Goals
Business Strategy
Model
Customer Relationship Manager rationale
BP Rental
Improve Customer Satisfaction
Attract new customers
Be rated higher than 6 by A C Nielson in EU (xg1.1)
To be a Premium Brand car rental company (SG1)
Opportunity
Room for competition (S3)
Be rated higher than 9 by A C Nielson (xg1.2)
Rentals requiring replace (xg3.2)
Support
Provide industry-leading customer service (SG2)
Customer Satisfaction survey (xg2.1) (xg3.1)
To provide well-maintained cars (SG3)
Model
Vehicle Manager rationale
BP Rental
Optimize Supplier Selection
Maximize ROI of cars
To have vehicles for rental when and where customers expect it (SG4)
Support

Figure 2.14: Alignment between decision maker goals and the overall strategy.

- Improved maintainability of the data warehouse and data warehouse models.
- Improved communication with users by means of partitioned diagrams.
- Alignment of the user requirements and the data warehouse with the business plan.

Nevertheless, there is still room for improvement. First, an automatic matching approach could be developed in order to aid the designer in matching data sources with the multidimensional models defined by user requirements. Second, the extension up to which business strategic models can be exploited is still unknown. Similarly, traceability information enables the use of techniques that could not be used before or were extremely time consuming, thus this could lead to additional extensions of the traceability model and new algorithms.

As a result, the work done in this PhD opens new potential lines of research.

1. First, an analysis of the relationship between business strategy models and the data warehouse. How the data warehouse can support more effectively the business plans and its evolution and, conversely, how changes in the business strategy affect the data warehouse.

2. Second, once we have identified that several decision makers are working towards the same business goal, how can they collaborate and interchange information in order to achieve better success.

3. Third, with the increasing number of data sources available worldwide, an interesting topic is to analyze how can data warehouses become more flexible to the addition of new data sources and how can these data sources be semi-automatically related to existing structures in the data warehouse schema.

4. Finally, fourth, the traceability support provided in this PhD Thesis allows to access any elements related to a given requirement at different abstraction layers. Thus, a new set of measures could be automatically calculated in order to guide the designer in the process of designing and implementing the data warehouse. Such set of measures could not only analyze the quality of the data warehouse, but also include estimations of the effort required in order to finish the implementation.

Bibliography

[1] N. Aizenbud-Reshef et al. “Model traceability”. In: *IBM Systems Journal* 45.3 (2006), pp. 515–526.

[2] G. Antoniol et al. “Recovering traceability links between code and documentation”. In: *IEEE Transactions on Software Engineering* 28.10 (2002), pp. 970–983.

[3] D. Barone, T. Topaloglou, and J. Mylopoulos. “Business intelligence modeling in action: a hospital case study”. In: *Advanced Information Systems Engineering*. Springer. 2012, pp. 502–517.

[4] A. Bitterer, K. Schlegel, and D. Laney. *Predicts 2012: Business Intelligence Still Subject to Nontechnical Challenges*. 2011. URL: `http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1873915`.

[5] A. Bonifati et al. “Designing data marts for data warehouses”. In: *ACM Transactions on Software Engineering and Methodology* 10.4 (2001), pp. 452–483.

[6] M.D. Del Fabro, J. Bézivin, and P. Valduriez. “Weaving Models with the Eclipse AMW plugin”. In: *Eclipse Modeling Symposium, Eclipse Summit Europe 2006*. Esslingen, Germany. 2006.

[7] X. Franch. “Incorporating Modules into the i* Framework”. In: *CAiSE*. Vol. 6051. LNCS. Springer Berlin, 2010, pp. 439–454.

[8] X. Franch, A. Maté, and J. Trujillo. “On the joint use of i* with other Modelling Frameworks: a Vision Paper”. In: *Proceedings of the 19th International Conference on Requirements Engineering*. IEEE, In Press.

[9] P. Giorgini, S. Rizzi, and M. Garzetti. “GRAnD: A goal-oriented approach to requirement analysis in data warehouses”. In: *Decision Support Systems* 45.1 (2008), pp. 4 –21.

[10] M. Golfarelli, D. Maio, and S. Rizzi. “The dimensional fact model: a conceptual model for data warehouses”. In: *International Journal of Cooperative Information Systems* 7.2 (1998), pp. 215–247.

[11] O. Gotel and S. Morris. “Macro-level Traceability Via Media Transformations”. In: *Requirements Engineering: Foundation for Software Quality.* Vol. 5025. Lecture Notes in Computer Science. Springer Berlin, 2008, pp. 129–134.

[12] Orlena CZ Gotel and CW Finkelstein. “An analysis of the requirements traceability problem”. In: *Proceedings of the First International Conference on Requirements Engineering.* IEEE. 1994, pp. 94–101.

[13] T. Hill and R. Westbrook. “SWOT analysis: it’s time for a product recall”. In: *Long Range Planning* 30.1 (1997), pp. 46–52.

[14] B. Hüsemann, J. Lechtenbörger, and G. Vossen. “Conceptual data warehouse design”. In: *Proc. DMDW.* 2000, pp. 3–9.

[15] W.H. Inmon. *Building the data warehouse.* Wiley-India, 2009.

[16] F. Jouault. “Loosely coupled traceability for atl”. In: *ECMDA-TW.* Nuremberg, Germany. 2005, pp. 29–37.

[17] F. Jouault and I. Kurtev. “Transforming models with ATL”. In: *Satellite Events at the MoDELS 2005 Conference.* Springer. 2006, pp. 128–138.

[18] R. Kimball et al. *The data warehouse lifecycle toolkit.* Wiley, 2011.

[19] A.G. Kleppe, J. Warmer, and W. Bast. *MDA explained: the model driven architecture: practice and promise.* Addison-Wesley Longman Publishing, 2003.

[20] James Kobielus et al. “Mighty mashups: do-it-yourself business intelligence for the new economy”. In: *Forrester Research* (2009).

[21] S. Luján-Mora, J. Trujillo, and I.Y. Song. “A UML profile for multidimensional modeling in data warehouses”. In: *Data & Knowledge Engineering* 59.3 (2006), pp. 725–769.

[22] A. Maté and J. Trujillo. “A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses”. In: *Information Systems* 37.8 (2012), pp. 753 –766.

[23] A. Maté, J. Trujillo, and X. Franch. “Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star”. In: *Journal of Systems and Software (In Press)* ().

[24] J.-N. Mazón, J. Pardillo, and J. Trujillo. “A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses”. In: *Advances in Conceptual Modeling Foundations and Applications.* Vol. 4802. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, pp. 255–264.

[25] J-N. Mazón and J. Trujillo. “An MDA approach for the development of data warehouses”. In: *Decision Support Systems* 45.1 (2008), pp. 41–58.

[26] J.-N. Mazón, J. Trujillo, and J. Lechtenbörger. "Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms". In: *Data & Knowledge Engineering* 63.3 (2007), pp. 725–751.

[27] Object Management Group. *Business Motivation Model (BMM).* 2010.

[28] Object Management Group. *The Meta Object Facility 2.0 Query/View-/Transformation.* 2005.

[29] R.F. Paige et al. "Building model-driven engineering traceability classifications". In: *ECMDA-TW* (2008), pp. 49–58.

[30] N. Prakash and A. Gosain. "Requirements driven data warehouse development". In: *CAiSE Short Paper Proceedings.* 2003, pp. 13–17.

[31] B. Ramesh and M. Jarke. "Toward reference models for requirements traceability". In: *IEEE Transactions on Software Engineering* 27.1 (2001), pp. 58–93.

[32] O. Romero and A. Abelló. "A framework for multidimensional design of data warehouses from ontologies". In: *Data & Knowledge Engineering* 69.11 (2010), pp. 1138–1157.

[33] C. Sapia et al. "Extending the E/R Model for the Multidimensional Paradigm". In: *Proceedings of the Workshops on Data Warehousing and Data Mining: Advances in Database Technologies.* Springer-Verlag. 1998, pp. 105–116.

[34] A. Sen and A.P. Sinha. "A comparison of data warehousing methodologies". In: *Communications of the ACM* 48.3 (2005), pp. 79–84.

[35] Alkis Simitsis, Panos Vassiliadis, and Timos K Sellis. *Extraction-Transformation-Loading Processes.* 2005.

[36] G. Spanoudakis and A. Zisman. "Software traceability: a roadmap". In: *Handbook of Software Engineering and Knowledge Engineering* (2005).

[37] N. Tryfona, F. Busborg, and J.G. Borch Christiansen. "starER: A conceptual model for data warehouse design". In: *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP.* ACM. 1999, pp. 3–8.

[38] P. Vassiliadis. "Data Warehouse Modeling and Quality Issues". PhD thesis. Athens, 2000.

[39] S. Walderhaug et al. "Traceability in Model-Driven Software Development". In: *Designing Software-Intensive Systems: Methods and Principle* (2008), pp. 133–159.

[40] S. Winkler and J. von Pilgrim. "A survey of traceability in requirements engineering and model-driven development". In: *Software and Systems Modeling* 9 (4 2010), pp. 529–565.

[41] R. Winter and B. Strauch. "A method for demand-driven information requirements analysis in data warehousing projects". In: *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on.* IEEE. 2003.

[42] E. Yu. “Modeling strategic relationships for process reengineering”. In: *Social Modeling for Requirements engineering* (2011).

[43] Y. Yu, J. Jurjens, and J. Mylopoulos. “Traceability for the maintenance of secure software”. In: *IEEE International Conference on Software Maintenance (ICSM 2008)*. IEEE. 2008, pp. 297–306.

Part II

PhD Thesis as a Collection of Papers

3

Including Traceability of User Requirements

The context of this chapter corresponds with the following papers:
Maté, A. *and Trujillo, J. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. Information Systems, (**IS**). 2012. Impact factor: 1.595*

One of the pitfalls in current data warehouse design approaches is the lack of traceability. Data warehouses support the decision making process by integrating information from several data sources. Thus, in order to take adequate decisions, the data warehouse must keep data fresh (relative to the context where the information is used). In turn, as the business environment changes, new information necessities arise, and decision requirements change. Thus, the data warehouse has to be updated in order to keep supporting the decision making process. Moreover, this means that changes in the data sources will affect the Extraction/Transformation/Load processes of the data warehouse and, ultimately, the data warehouse schema.

Since most approaches focus on designing a data warehouse from scratch, they pay little attention on how a change affects the data warehouse. In turn, the synchronization between the different diagrams is lost when performing several changes, thus traceability from user requirements and data sources towards the data warehouse is also lost. This transforms what should be relatively simple tasks, such as evaluating if a new requirement incurs into gathering new data, into a complex task requiring to carefully analyze the whole structure of the data warehouse. This process becomes even more complex as the size of the data warehouse grows.

In order to avoid this problem, the first part of this PhD Thesis focuses on the definition of a traceability metamodel, that provides end-to-end traceability between the different data warehouse models, allowing us to maintain the complexity of these tasks constant and diminish the cost of introducing a change into the data warehouse.

Information Systems 37 (2012) 753–766

Contents lists available at SciVerse ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses

Alejandro Maté*, Juan Trujillo

Lucentia Research Group, Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n - 03690 San Vicente del Raspeig - Alicante, Spain

ARTICLE INFO

Available online 11 May 2012

Keywords:
Data warehouse
Business Intelligence
Traceability
User requirements
MDA

ABSTRACT

The complexity of the data warehouse (DW) development process requires to follow a methodological approach in order to be successful. A widely accepted approach for this development is the hybrid one, in which requirements and data sources must be accommodated to a new DW model. The main problem is that we lose the relationships between requirements, elements in the multidimensional (MD) conceptual models and data sources in the process, since no traceability is explicitly specified. Therefore, this hurts requirements validation capability and increases the complexity of Extraction, Transformation and Loading processes. In this paper, we propose a novel trace metamodel for DWs and focus on the relationships between requirements and MD conceptual models. We propose a set of Query/View/Transformation rules to include traceability in DWs in an automatic way, allowing us to obtain a MD conceptual model of the DW, as well as a trace model. Therefore, we are able to trace every requirement to the MD elements, further increasing user satisfaction. Finally, we show the implementation in our Lucentia BI tool.

1. Introduction

Data warehouses (DW) integrate several heterogeneous data sources in multidimensional structures (i.e. facts and dimensions) in support of the decision-making process [1,2] in Business Intelligence. Therefore, the development of the DW is a complex process which must be carefully planned in order to meet user needs. In order to develop the DW, three different approaches, similar to the existing ones in Software Engineering (bottom-up or supply-driven, top-down or demand-driven, and hybrid), were proposed [3,4].

The first approach, supply-driven, follows a bottom-up process and makes use of the information in the data sources while ignoring the user requirements. As the schema is not adapted to the user needs [3], the DW may fail to meet the user expectations. The second approach, demand-driven, follows a top-down process and focuses on the user requirements while ignoring the data sources. Therefore, it is possible that some of the user needs cannot be satisfied because the necessary data has not been stored [4]. The third approach, hybrid, makes use of both data sources and user requirements [5]. Therefore, with this approach user requirements which cannot be satisfied are noticed in earlier stages. Once the information from both worlds is collected, the incompatibilities have to be solved by accommodating both data sources and requirements in a single model.

However, with the hybrid approach a new problem arises. In the top-down and bottom-up [3] approaches,

* Corresponding author. Tel.: +34 96 5909581x2737; fax: +34 96 5909326.
E-mail addresses: amate@dlsi.ua.es (A. Maté), jtrujillo@dlsi.ua.es (J. Trujillo).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.05.003

every element used for the implementation of the DW comes from a single source only (either requirements or data sources), thereby allowing us to trace elements by name matching. On the contrary, in the hybrid approach, DW elements must be accommodated to consider the information from both sources. Therefore, they are merged into hybrid elements containing information from both worlds. In turn, additional effort is required in order to check which parts of the DW match, not only with each requirement, but also with each part of the data sources. As user requirements are specified using in business terms [6] (i.e. "Professors", "Students", "Number of professors"), while data sources are composed of tables, files and other heterogeneous sources of data, they rarely match with each other (i.e. table "tt_pers" containing information about both professors and students, or "Number of professors" not existing at all in the data sources). Indeed, as pointed in [6] and supported by our experience, by following the hybrid approach, changes to accommodate elements are done almost in every project, since it is very common that user requirements and data sources do not match, thus losing the implicit traceability.

In this process, the relationships between the elements are not recorded and lost, since there is no explicit traceability included in the development process. In turn, this hurts requirements validation [7–9], making it unable to check the current status of each requirement or take decisions about alternative implementations if a given requirement cannot be fulfilled. Although the traceability aspect has been thoroughly studied [8,10–17], it has been almost completely overlooked in DW development. To the best of our knowledge, references to requirements traceability in DW development approaches are those from [18], which only mention implicit traceability by name matching. Other approaches [2,3] do not consider requirements traceability in their methodology as a first class element, with the exception of [19], where requirements traceability is considered as a quality measure for DWs. In these approaches, mappings are created when successive models are derived, however no automation or semantics are provided and no approach to maintain mapping consistency is described in case that models are modified.

Although several trace metamodels have been proposed in literature, due to the special idiosyncrasy of DW development, a trace metamodel specifically tailored to face several challenges is required: (i) connecting multiple sources with multiple targets in a meaningful way, as requirements need to be reconciled with data sources which may, or may not, match the expectations of the users. As opposed to traditional information systems, in DW domain we must trace indicators and goal elements from requirements up to tables, which may be facts or dimensions, and must be differentiated in order to reason about their relationship. (ii) Not relying on element types and being able to retrieve them from their respective models. As currently there is no standard for requirements nor multidimensional modeling in DWs, the trace metamodel must be prepared to link different traced models with minimum effort, and (iii) given the complexity of DW models, minimizing the overhead introduced in the development process with the inclusion of traceability, by defining how traces should be generated in an automatic way, and maintaining them without user intervention wherever possible.

In our previous work [5,18,20,21], we defined a hybrid DW development approach in the context of the model driven architecture (MDA) framework [22]. MDA is suitable for the development of the DW, since DW models may present a high number of elements which must be derived into the next layer. For example, our projects present between 50 and 80 elements at requirements level per model, which otherwise would be manually derived to the conceptual and logical models. Therefore, automating the process saves time and effort for the developers. In our approach, requirements are specified in a Computation Independent Model (CIM) by means of a UML profile [18] based on the i* framework [23]. Then, they are automatically derived, reconciliated with the data sources in a hybrid model, and refined through a series of layers (Platform Independent Model (PIM) layer

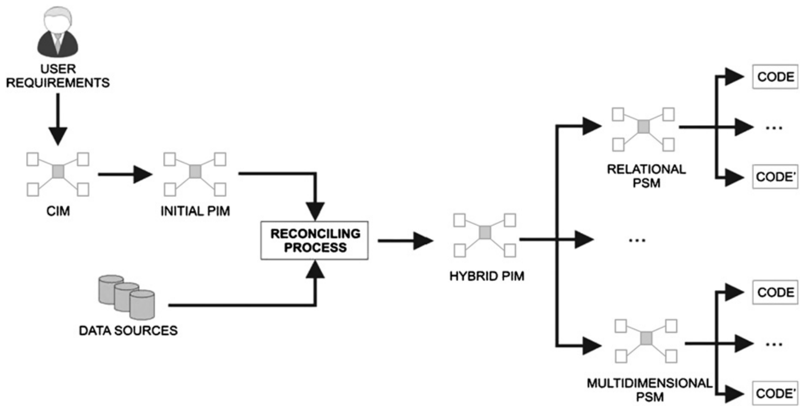


Fig. 1. Layers in our DW approach.

and Platform Specific Model (PSM) layer) until the final implementation is achieved, as seen in Fig. 1.

The automatic derivation is done by means of model to model transformations specified by Query/View/Transformation (QVT) [24] rules. QVT is a language defined by the Object Management Group (OMG) and proposed as standard to create model to model transformations. However, due to our experience in real-world projects, the lack of traceability does not allow us to adequately validate requirements due to the mismatch between user requirements and data sources, and incurs in additional costs when requirements change.

In [25], we complemented our previous works with the inclusion of a novel traceability metamodel for DWs and an automatic derivation of the corresponding trace models. In this paper, we provide the complete specification of all the necessary transformations to fully automate the generation of CIM to PIM traces. Furthermore, we provide the description of how these transformations have been implemented in the Lucentia BI tool. In this way, by including traceability, we improve the reusability, maintainability and rationale comprehension of the models [7,9], and we are able to easily analyze which requirements have been met and which elements from the models will be affected by a change in a given requirement. One of the great advantages of our proposal is that it can be implemented in a tool, allowing us to automate the full process, cutting costs and time which would be invested in trace recording.

The rest of the paper is structured as follows. Section 2 presents related work about traceability. Section 3 introduces our traceability metamodel for DWs and the inclusion of trace models in our approach. Section 4 presents the QVT rules for automatic derivation of traces in the DW context, focusing on the derivation of requirements into a conceptual DW model. Section 5 presents an example of application, in order to show the benefits of our proposal. Section 6 presents the implementation of the proposal in the Lucentia BI tool. Finally, Section 7 outlines the conclusions and sketches the future work to be done in this area.

2. Related work

In this section, we will discuss the existing traceability research in other fields, its benefits and problems, and we will also discuss its current status in the DW field. Currently, traceability can be studied from two different points of view. The first one is the requirements engineering (RE) field, whereas the second one is the model driven development (MDD) field. Although both fields are focused in different aspects of traceability, they also present common issues.

Most of the work done until now has been in the RE field [8,10,14–17,26,27]. Some authors [9,15] consider pre-requirement specification (pre-RS) as a more complex scenario, since it has to deal with artifacts written in natural language and different points of view (i.e. stakeholders and developers), and post-requirement specification (post-RS) as a simpler one since the requirements are already modeled.

The main benefits provided by traceability have been studied in this field [7,14,17]. Traceability helps assessing the impact of changes and rationale comprehension, by identifying which parts of the implementation belong to each requirement [14]. It also helps the reusability and maintainability, since the scope of each part of the project is known and defined thanks to the traces. In turn, these benefits help lowering the costs associated with the project [7,17].

The main drawbacks mentioned about traceability are the non-existence of a standard traceability definition or metamodel, the manual recording of traces, and that traceability itself is seen as a burden until it is necessary later on although its benefits have been tested [7]. This situation creates a problem which makes difficult to successfully apply traceability.

In order to alleviate the first drawback, a classification of eight categories for traces was presented in [8] in order to provide a common language, and some metamodels focused on pre-RS have been proposed [17]. However, despite these efforts, none of these metamodels has been acknowledged as a standard. Furthermore, given the recent rise in MDD and goal-oriented models [3,10–12,18,28], some of the concepts included in these trace metamodels, such as “Stakeholder” [17] or “User” became redundant, as MDD approaches incorporate models which are already consider these concepts as first class elements among their constructs. In addition, as these metamodels do not cover post-RS traceability on detail, they are not able to establish meaningful relationships between design elements.

The second and third drawbacks can be solved by automating the trace recording. Nevertheless, in the RE field, the trace recording is focused on pre-RS traceability, and needs to find traces in documents in natural language. In turn, this generates models that must be supervised, with a high percentage of irrelevant traces that complicate the comprehension and visualization of the trace model, which usually has a huge number of traces already [29].

On the other hand, in the MDD field, the MDA framework is used [10–13,29]. The automatic derivation process starts from a CIM layer, where the requirements are specified as models, usually by means of goal-oriented models [3,18,28,30–33]. In this way, the traceability research in the MDD field is mainly focused on post-RS, which makes the automation of traces an easier task and less prone to errors, since everything is either a model or an element in a model. However, although more restrictive, the traceability definitions in this field are not standard either. As such, multiple metamodels have been proposed to address traceability issues [34–38] following an MDD approach for software development. Some approaches are focused on specific aspects such as recording automatic transformation traces [34] or maintaining trace consistency [35], while others are aimed to provide support to include traceability for the whole development process [36–38]. In [37], a generic approach using trace tagging is proposed. The benefit of this proposal is that it can be easily adapted to different domains, as semantic tags are considered unrestricted strings attached to a trace. However, it relies on the designer to specify the different trace concepts. As different users make use of different concepts, this approach may hinder the interoperability between tools and communication between users, as was shown in [7]. In [38] the authors propose a metamodel which includes typed traces which relate pairs of artifacts. However, this kind of metamodel lacks (i) flexibility when one or more source artifacts are to be related with one or more target artifacts simultaneously, for example a requirement being related to a set of conceptual elements, and (ii) adaptability when the types of the artifacts to be related are unknown due to the absence of a standard, such as in the case of DWs. In [36], the first drawback is solved by allowing *n* number of sources and targets to be related by the trace links, although the second drawback still exists.

In addition to the different metamodels, there are mainly two definitions of traceability in the MDD community. The definition we will use in this paper comes from [13]; They define traceability as "[...] the ability to chronologically interrelate uniquely identifiable entities in a way that matters. [...] [It] refers to the capability for tracing artifacts along a set of chained [manual or automated] operations."

In the DW field, as we previously stated, there is no mention of traceability being included in the process, even though there are approaches which would benefit from it. These approaches are based on model transformations through multiple layers, either following MDA [5] or a similar set of layers [19]. Currently, whenever a change to an element is done, the traceability as defined in [13] is lost, since the elements are associated by name matching. Therefore, we lose all the aforementioned benefits of traceability, which can be obtained in the DW field at a low cost, since trace recording can be automated. Moreover, the quality metrics presented in [19], such as the *correctness* of the DW (conflicts between the specification and the real world, i.e. in our case, conflicts between entities/dimensions as specified by requirements and their real structure), could be provided in a more automated way with traceability support, as opposed to performing the process manually, thus allowing us to increase the quality of the final implementation.

Due to the peculiarity and idiosyncrasy of data warehouses, we will need to differentiate between (i) the traces coming from the requirements (for requirements validation and impact change analysis), (ii) the traces coming from the data sources (for querying and derivation of initial Extraction, Transformation and Load processes) and (iii) the traces linking elements in the multidimensional conceptual models, based on their particular relationships [20].

3. A traceability approach and a trace metamodel for data warehouses

As previously stated, if we wish to perform automatic operations with traces, we must be able to identify the meaning of each trace. In order to do this, we need to elaborate a set of trace types, which define the semantics of the relationships between elements. In this section, we will introduce the trace metamodels proposed in the MDD field along with our proposed metamodel for DW.

3.1. Model driven architecture metamodels for traceability

Our traceability approach is based on the trace framework proposed by the OMG, which is included in the MDA framework [22].

The metamodel, presented in Fig. 2, is composed by a *transformation record* which represents the transformation that generated the traces. The transformation record contains the set of traces produced and can have associated metadata as, for example, the parameters passed to the transformation when it was executed. For each trace recorded, there is a set of model elements, referenced as *objects*, which are linked by the previously mentioned trace, varying from 0 to N elements. As in the previous case, the trace can have associated metadata as, for example, the rule of the transformation that created each trace.

According to this proposal there is a core metamodel for the ATLAS Model Weaver (AMW) [39], used for linking elements from models. This core metamodel constitutes the base for the traceability metamodel which we extend.

3.2. Proposed metamodel

Our proposed metamodel for traceability extends AMW metamodel for traceability, including the necessary semantic types for traceability in DW. The result can be seen in Fig. 3.

In this metamodel, a *TraceModel* has a set of models (*wovenModels*) linked by the trace model. Each of these woven models has the list of references (*ElementRef*) which identify the elements linked by the traces. The trace model has also a set of *TraceLinks*, which define the relationships between the elements in the woven models. Each trace link has a set of *sourceElements*, which were the source of the automatic derivation, and a set of *targetElements* which were the result of the automatic derivation. A trace link can also have one parent, as well as a set of children trace links. This feature is provided by the base class in the AMW model *WLink* (see [39]). This is an important feature, since it allows us to group traces forming hierarchies, providing different levels of detail in the trace models. By following this approach, we improve the visual scalability of the trace models, allowing the user to analyze them without being

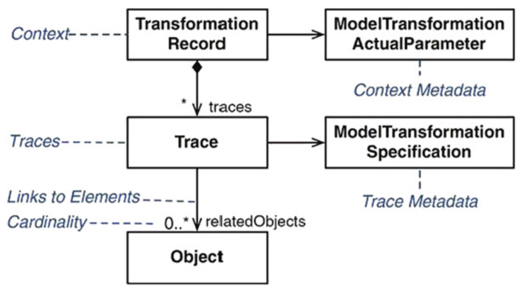


Fig. 2. Metamodel for traceability in MDA.

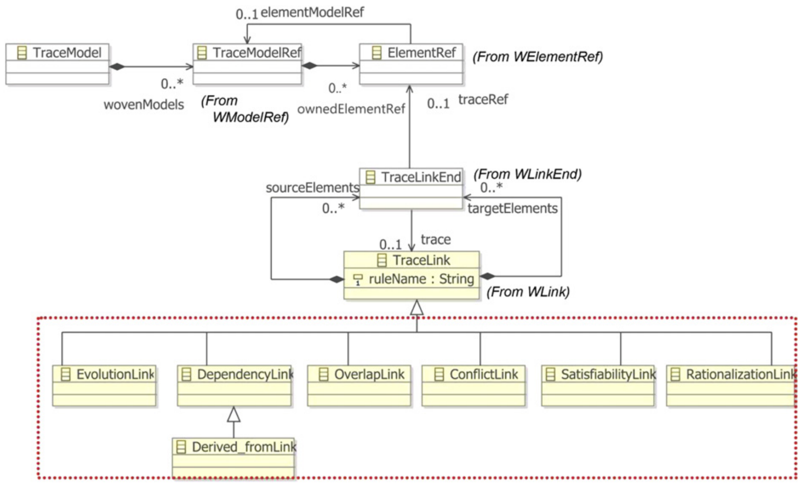


Fig. 3. AMW metamodel for traceability extended with semantic links for DWs.

overwhelmed by the complexity and hiding the undesired details. The elements linked by the traces are represented by the *TraceLinkEnds*, which reference the identifiers listed in the woven models.

In order to add semantics to the traces in the metamodel, we extend the *TraceLink* element, aligning the types with the classification made in [8]. We could use a reduced set of links, since in our case *Overlap* and *Conflict* are very similar. However, for the sake of standardization, we include the relevant set of links as-is. Nevertheless, in our case, each trace will only have one semantic type attached (since we do not include roles because they are included at the CIM level). Therefore, the definition of each trace link type is as follows:

- *Satisfiability* and *Dependency* will be used for vertical traceability (between different layers). In the first case, the traces with this type will be those coming from the requirements (in the CIM layer) to the elements in the PIM. In the second case, we will use a specialization of the *Dependency* type, *Derived_from*, in order to specify the traces coming from the data sources to the multidimensional elements at the PIM level.
- *Evolution* links will be included to handle horizontal traceability which takes care of element changes at the same layer (e.g. from PIM to PIM).
- *Overlap* and *Conflict* will be used for solving conflicts where the same element comes both from the requirements and from the data sources in a different shape. In this case, the designer will decide which derived element is the correct solution to the conflict.
- *Rationalization* links will be included as means of enabling the user to record his own annotations in the trace model about changes or decisions taken.

In addition to these links, the initial classification also included *Contribution* and *Refinement* links. Contribution

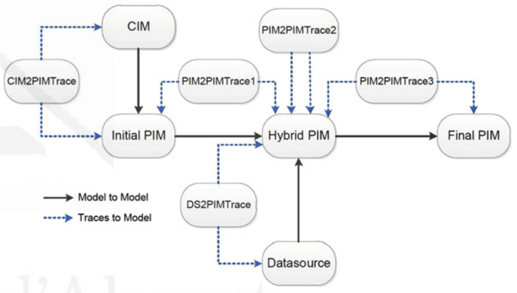


Fig. 4. Inclusion of traceability models in DW development process.

links were omitted, since they link stakeholders to requirements. As stakeholders are already modeled in DWs by using goal-oriented approaches, this semantic link became redundant. On the other hand, Refinement links, which disaggregate complex artifacts into simpler ones, were omitted since currently there is no artifact in the development process which could benefit from its application. Future applications for this semantic link may arise when current development approaches are refined.

Once we have defined our trace metamodel, we need to define an approach to create the trace models in an automatic way, which models will be created and what information will they store. In order to include traceability in our approach [5,18], we will introduce the trace models shown in Fig. 4. The first step to include traceability and support for automated operations (like requirements and transformation validation, calculation of traceability measures and derivation including source datatypes) in our approach is to make the relationships (shown in [5,18]) between the CIM and PIM elements explicit. This is a vertical traceability (source and target models are in a different MDA layer) case in which all the relations are perfectly known, since they are created in an automatic way, so we just need to create the elements which correspond to the traces simultaneously as the transformation is executed and the target model is created. This CIM2PIMTrace model will be storing mainly *Satisfiability* traces.

The second step is to be able to record the traces between the data sources represented in the PSM and the hybrid PIM. The hybrid PIM is the result of a transformation using as input the first PIM (from now on "initial PIM") and the data sources. The hybrid PIM should be traced both to the initial PIM, in order to be able to trace the original requirements, and to the data sources, in order to keep track of the source tables and attributes. This hybrid PIM can contain conflicts between concepts that come from both the requirements and from the data sources, either defining the same concept differing only in their name (overlap) or totally differing both in name and attributes (conflict). In this way, this hybrid PIM will have both vertical (between the data sources and the hybrid PIM, DS2PIMTrace) and horizontal (between the initial PIM and the hybrid PIM, PIM2PIMTrace1) traceability models. The vertical trace model between the PSM and the hybrid PIM will record the *Derived_from* traces, whereas the horizontal trace model between the initial PIM and the hybrid PIM will record the *Evolution* traces. An additional PIM2PIMTrace2 model can be added to record the existing *Overlaps* and *Conflicts*, but these should be manually added, since only the designer knows which elements in the model refer to the same concept. It is important to note that, these kind of traces, are not less important than the fully automatic ones, since they will act as a bridge to map certain requirements to the data sources. Since this task is related to matching sets of elements with the same semantic meaning, recent research on topic modeling for traceability [40] could be applied, in order to aid the designer with this task, making the process semi-automatic and reducing the time costs.

The last step is deriving the final PIM, which will be used to generate the target DW. This final PIM retains only the elements from the hybrid PIM which will be finally used. In this way, this PIM is the result of filtering the undesired elements and resolving the conflicts which appeared in the hybrid PIM. Therefore, the traces from the hybrid PIM to the final PIM will show which concepts were chosen as a solution to each existing conflict. The type of these traces will be *Evolution* and will be stored in the PIM2PIMTrace3 model.

Since the development process is performed by successive deriving, adding, and filtering elements, while most elements are not altered, the traces have low volatility. In this way, our current development of a reactive framework which automatically updates the corresponding traces whenever a change (update or delete) is made minimizes the maintenance effort. Further details are provided in the next section.

Once we have defined the trace metamodel and we have shown all the required trace models, we need to formally define the automatic derivation of the traces by means of QVT [24] rules. In order to illustrate this process, and given the complexity of DW development, we will focus on the generation of traces from CIM to PIM models in the next section.

4. Automatic derivation of traceability models in data warehouses

In this section, we will discuss the necessary transformations to automatically generate the aforementioned traces and store them in trace models, which can be updated over time. In order to automatically generate traces along with DW models, we will need to define a set of transformation patterns. Given that DW development is a complex process, the whole set of patterns for each step cannot be included in a single paper, thus we will focus on the traces coming from CIM to PIM, both in this section and in our example. In the case we wanted to apply our proposal to other set of DW development approaches, the transformation patterns should be redefined accordingly to the modeling technique used, although the trace metamodel would not require any modifications.

According to our proposal for developing DWs [5,18], we use a hybrid approach deriving the elements in an initial PIM model from the requirements by means of QVT rules. QVT rules specify a transformation by checking for a defined pattern in the source model. Once the pattern is found, a QVT rule transforms elements from the source metamodel into the target metamodel. A QVT, which creates the links between the CIM *business process* element and the PIM *star package*, *fact package* and *fact* elements, is shown in Fig. 5.

On the left side of the transformation rule is the source metamodel, which in our case, is our i^* profile for modeling requirements for DWs. In this QVT rule, we have a business process with its associated rationale, modeled by means of strategic, decisional and information goals. These goals model the business logic, from which we obtain the information requirements. In turn, information requirements are decomposed into measures (indicators of business performance), and contexts.

On the right side of the transformation rule are the target metamodels. On the one hand, in the upper-right side, we have our multidimensional profile for DWs. The elements generated for this transformation are: (i) the fact, which is the focus of the analysis (i.e. "educate", "sales"), related to the business process, (ii) the fact package, which is provided to increase the modularity and scalability of the system, and includes the fact and manages the relationships with the dimension packages, and (iii) the star package, which serves as a second level of modularization, including the fact package and the different dimensions packages. On the other hand, in the lower-right side, we also have the trace metamodel we proposed, composed in this case by the satisfiability link which makes explicit the relationship between the business process and the star package, the fact package, and the fact.

The "C" at the center of the figure means that the source model is checked, whereas the "E" means that the target models are enforced. This means that, each time that the described pattern is found in the source models, the target patterns are enforced (generated) in the resulting target models.

Finally, we have a set of where clauses, which specify actions which cannot be modeled in the declarative part. In our case, these actions are related to retrieving and specifying the values of the different variables, as well as

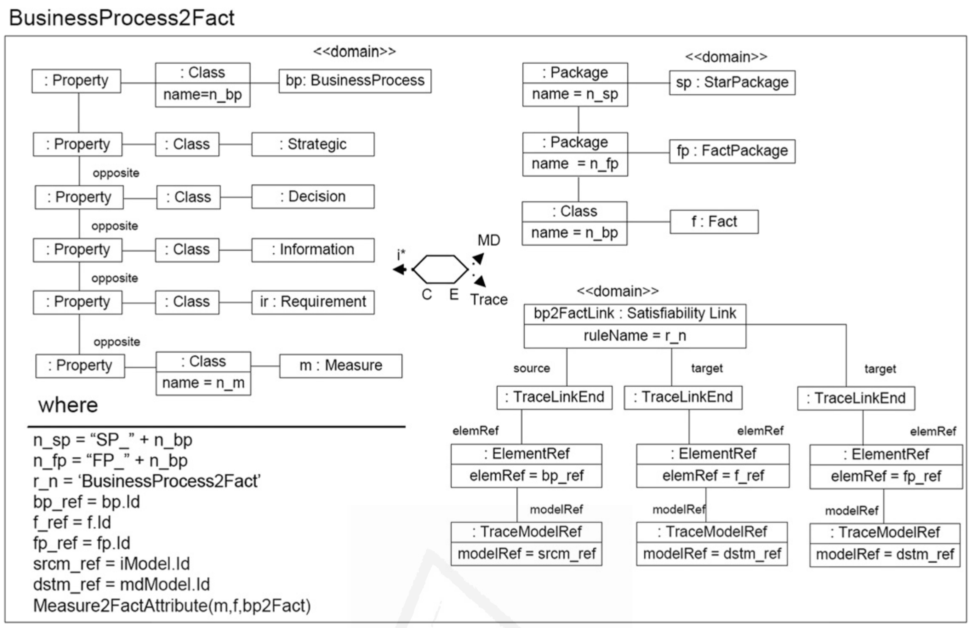


Fig. 5. QVT rule to derive a fact from a business process along with the associated trace.

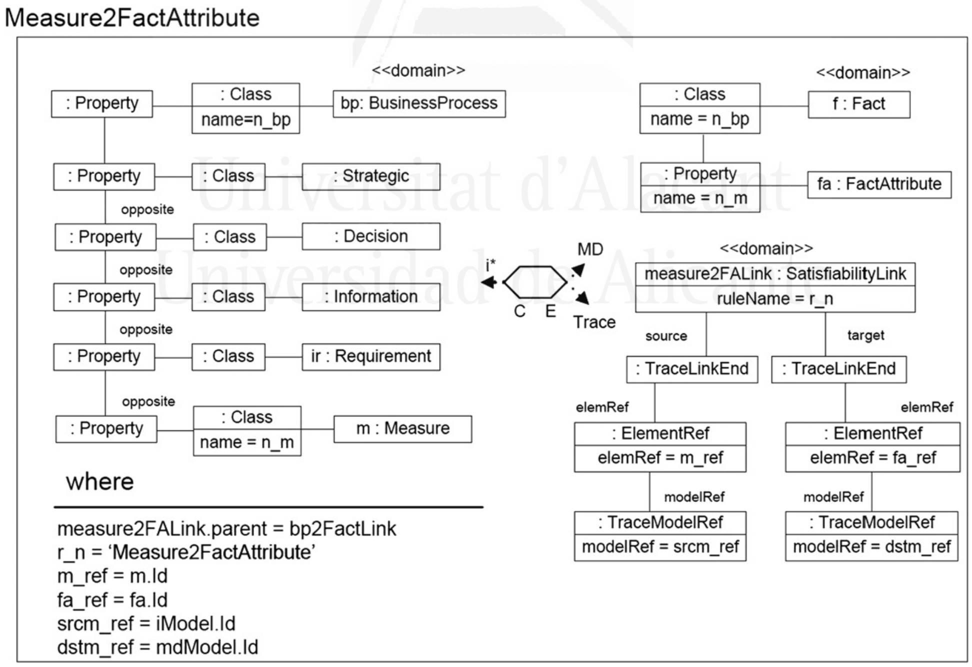


Fig. 6. QVT transformation for deriving a measure into a fact attribute with its associated trace.

calling the QVT rule for mapping the corresponding measure into a fact attribute (Measure2FactAttribute). This QVT rule is shown in Fig. 6.

In this figure, we obtain a *fact attribute* at PIM level from a *measure* at CIM level. A measure, at CIM level, is an atomic indicator which provides information about the performance of the business process. For example, we can consider "grades" for the "educate" process. The source metamodel pattern is the same as in the previous case, since we are calling this rule from the Business2Fact transformation. However, the generated models are different. Now, in the upper-right side, we can see a fact attribute, which is a column in the fact which will be aggregated by the different dimensions, and is related to a measure at the CIM level. This fact attribute will be included as a property of the previously generated fact, which is passed as a parameter. On the other hand, in the lower-right side, we have our trace metamodel. In this case, we generate a satisfiability link which links the corresponding measure at CIM level with the fact attribute.

Finally, a particularly interesting part of this transformation is located in the "where" clause. In addition to the specification of values for different variables, in this transformation we specify a hierarchical relationship between trace links. In this way, "measure2FALink" becomes a child of "bp2FactLink" connecting the business process to the fact. The reason to specify this hierarchical structure is twofold. On the one hand, we reduce the visual complexity of trace models, as specified in Section 3. On the other hand, the parent–child relationship between trace links reflects the relationship between multidimensional elements, where the fact attribute is actually included as a part of the fact. Therefore, this structure helps us when analyzing the impact of changes over the models.

After deriving the fact and the fact attributes from the CIM model, we proceed to obtain the different *dimensions* for the DW. Whereas the fact constitutes the focus of analysis, dimensions represent the context of analysis (i.e. "student" or "subject"). Dimensions are generated from contexts at CIM level, which are entities that provide relevant information about the fact ("product" or "market"), and constitute an important aspect for DWs because their *levels* can be structured into *hierarchies*, which specify partially ordered sets of levels. Therefore, they provide different aggregation levels to analyze the data (i.e. "individual subjects" or "subjects by category"). These different hierarchies are grouped into a *dimension package* in the models, hiding the complexity of a given dimension unless it is necessary (i.e. "person" having a hierarchy for city, state, country and another hierarchy for age segments). The generation of dimensions and levels is separated into two QVT rules. The first rule, transforms a context from the CIM into a dimension in the PIM. This is the case when the context is not the result of an aggregation from any other context. The QVT rule for transforming a context can be seen in Fig. 7.

In this figure, a context "c" from the CIM, on the left side, is transformed into (i) a dimension package "dp", (ii) a dimension "d", which will be related to the fact, and (iii) its associated base level "b" in the PIM, at the right side of the figure. The associated satisfiability link to this transformation has a single source element reference, which is the context from the CIM, and four (two omitted) target trace link ends. Each trace link end references a different element but has the same model reference (since the elements are in the same target model). The omitted trace link ends correspond to the dimension package, and the association between the base level and the dimension.

Finally, the generation of *dimension hierarchies* is performed by successively matching aggregation *contexts* at CIM level, which shape the dimension hierarchies at PIM level. The QVT rule for generating dimension levels is

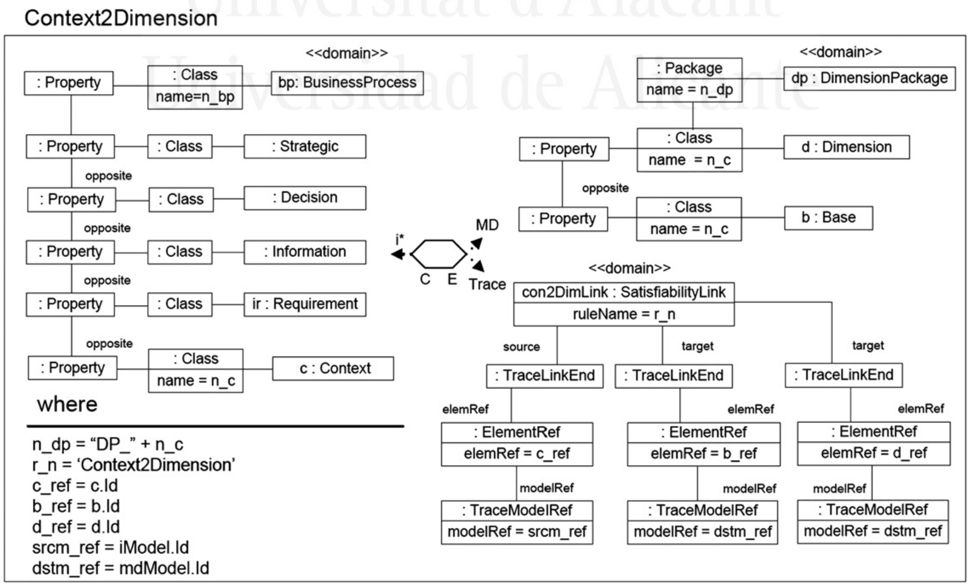


Fig. 7. QVT transformation for deriving a context into a dimension with its associated trace.

shown in Fig. 8. In this figure, we have two contexts "c", and "c1'", from the CIM level on the left side of the figure, which are related by an aggregation relationship. These two contexts are transformed into two bases "b" and "b1" at PIM level, as well as a "roll up" relationship between them, which specifies that instances corresponding to level "b" can be aggregated into instances at level "b1".

Since the source pattern is a generic one, i.e. the context "c" could possibly match this rule, as well as the Context2Dimension rule, we are focusing on (i) the generation of the aggregated level "b1", and (ii) establishing the corresponding "roll up" relationship between levels. After generating the level and roll up relationship elements, they are packaged into the corresponding dimension package, as specified in the "where" clause. This process will be repeated for each pair of contexts which present an aggregation relationship, thus generating all the dimension levels and roll up associations between them. Therefore, we will only generate one satisfiability link per execution of the rule. This satisfiability link takes as a source "c1", and as target elements "b1" and the roll up association, thus avoiding the generation of different redundant trace links between the root context "c" and the root level "b" at the PIM level.

In an analogous way as how the fact attributes were part of the fact, dimension hierarchies are part of a dimension, thus the satisfiability links relating hierarchy levels are considered children of the satisfiability link related to the dimension. Since a typical data mart may have between 15 and 20 dimensions at most [2], this structure results into simple trace models which can be easily navigated by the user if he wishes to explore the trace model. Some examples of this structure are shown in Section 6.

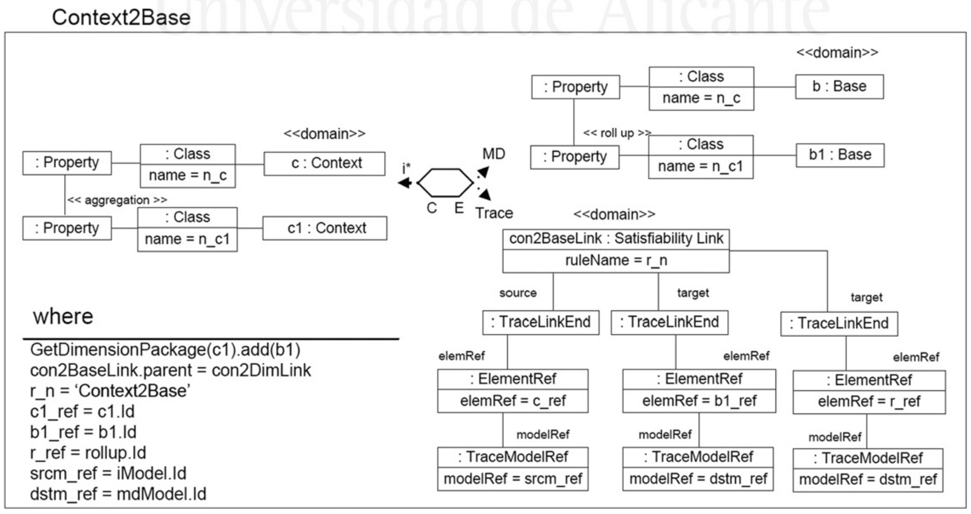
Finally, after having presented all the necessary QVT rules, we summarize the relationships presented in this section between the CIM and PIM elements in Table 1, along with its corresponding rule name.

As we can see, most elements at CIM element generate a number of different elements at PIM level. In the specific case of contexts, these elements can derive either into a dimension package, its corresponding dimension and first base level, into a level and a roll up association with the previous level in the dimension hierarchy. In order to provide a better overview, a graphic example of the generic relationships is shown in Fig. 9.

In this figure, we can see the previously described elements in the QVT from the CIM at left side. The business process has its associated rationale represented by means of successive goals, which derive into an information requirement for the DW. In turn, this information requirement is decomposed into contexts and measures. At the right side of the figure, the corresponding multidimensional elements from the PIM appear. As aforementioned, the business process is related to the fact (and the corresponding packages), whereas the first context is related to the dimension package, the dimension, the base level and its association with the dimension. On the other hand, the second context is associated with the second level in the dimension. With this approach, if we

Table 1
Relationships between CIM elements and PIM elements.

Rule name	Source element (CIM)	Target element (PIM)
BusinessProcess2Fact	BusinessProcess	StarPackage
BusinessProcess2Fact	BusinessProcess	FactPackage
BusinessProcess2Fact	BusinessProcess	Fact
Measure2FactAttribute	Measure	FactAttribute
Context2Dimension	Context	DimensionPackage
Context2Dimension	Context	Dimension
Context2Dimension, Context2Base	Context	Level
Context2Dimension, Context2Base	Context	Association



GetDimensionPackage(c1).add(b1)
con2BaseLink.parent = con2DimLink
r_n = 'Context2Base'
c1_ref = c1.Id
b1_ref = b1.Id
r_ref = rollup.Id
srcm_ref = iModel.Id
dstm_ref = mdModel.Id

Fig. 8. QVT transformation for deriving aggregation contexts into dimension levels with their associated trace.

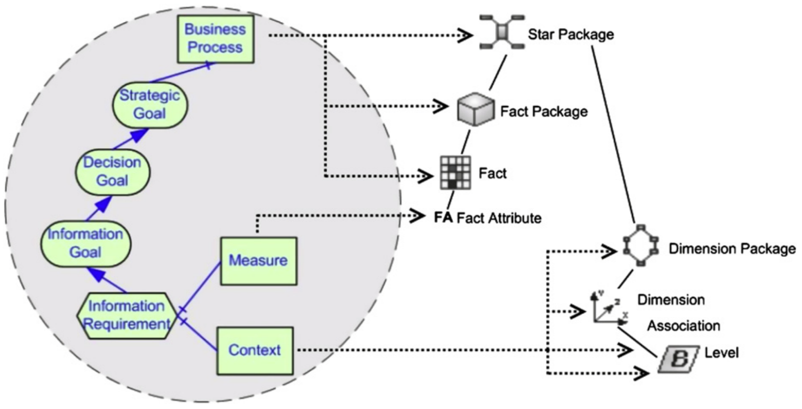


Fig. 9. Generic relationships between CIM elements and PIM elements.

wish to check the result of the transformations for debugging, we can check which rule created each element with the information stored in the traces. In addition, these traces allow us to keep track of which elements in the PIM model match with each requirement in the CIM model, making requirements validation easier. Moreover, if any requirement is changed, we know which elements are affected and which rule created them, being able to execute the corresponding rule of the transformation to regenerate the affected part of the PIM.

Once traces have been generated, we have to consider how traces will be maintained. As we previously introduced in Section 3, given the low volatility of traces in our approach, a simple reactive framework can minimize the effort invested in maintenance of traces. Although the details of this framework are out of the scope of this paper, we provide a brief description of its implementation. There are three possible operations to be performed over DW models which we can consider: addition, update, and delete, and two possible kinds of traces according to the direction in the methodology: forward and backward. On the one hand, additions are automatically managed by a change propagation algorithm. On the other hand, updates and deletions are managed by the reactive framework and the change propagation algorithm.

First, we have additions. Additions represent a refinement of the model by the designer. In this way, they may or may not be related to a given requirement. In this sense, no new traces are included backwards (i.e. from PIM to CIM, unless stated otherwise by the DW designer). However, if a new element is added, new forward traces are generated when propagating changes to the derived models. For example, we add a new attribute ''surname'' to the base level ''Student'' in the first PIM, and it is propagated and included into the derived PIM models. Second, we have deletions. Deletions represent a simplification or discard of certain elements. Deletions are managed in one part by the propagation algorithm and in other part by the reactive framework. When a delete operation is performed over an element, its related backward traces (i.e. from PIM to CIM) and those of the deleted children are removed from the trace model, since the elements which satisfied those traces no longer exists. Using the previous example, the ''Student'' level is deleted, therefore its associated traces are removed, since no element satisfies the context ''Student'' at requirements level any longer. Afterwards, when propagating changes through a change propagation algorithm, forward traces no longer having a source are deleted, along with its corresponding elements. For example, the forward traces which connect the ''Student'' level and its attributes with their counterparts in the derived PIM models are then deleted along with the model elements. Finally, we have update operations. Update operations may include renaming and structural changes. When an element is renamed, the reactive framework updates the name of the corresponding trace link ends in the trace models. Although these names are not used by the change propagation algorithm or the framework, they provide an easy way for the designer to identify and analyze the trace model. For example, if we rename ''Student'' to ''Customer'', the names shown in the trace model would be updated, although their id remains the same. On the other hand, structural changes include the addition or deletion of other elements, which may influence how elements are matched with the corresponding QVT rules. In these cases, the change propagation algorithm re-structures the target model accordingly. Using this approach, traces are maintained in an automatic way, thus minimizing the effort required to apply traceability.

Although in this section we have presented the generation of traces from a goal-based CIM, the traces could be used to trace any element in a different CIM model from another proposal, as long as it has a unique reference identifier. Therefore, the effort required to apply our proposal is limited to the cost required to add the trace generation code into the existing QVT rules.

5. Example of application

In this section, we will present an example of application for our traceability proposal, showing how the traces can be navigated to retrieve useful information.

A University wishes to build a DW in order to analyze the factors which influence the performance of the students. This university has a transactional database created for managing the information about professors, degrees, subjects and students, which will serve as data source for the data warehouse. The analysts wish to analyze the *students grades* by *subject*, *professor* who teaches them, and *student*, taking into account how many *hours* they spend studying per week. These requirements are recorded in a CIM which acts as the starting point for the process.

In Fig. 10, we can see the requirements on the left side, where the business process (focus of the analysis) in this case is the student success. Its corresponding contexts are the students and the students grouped by hours of study per week (shown in the figure), the subjects and the professors (omitted due to space constraints). On the right side of the figure, we can see the PIM with the StarPackage ''SP_student success'', the FactPackage ''FP_student success'' and the Fact ''Student success'', associated with the business process. The measure ''student grades'' is associated with its corresponding FactAttribute, whereas the ''student'' context derives into the DimensionPackage ''DP_Student'', the Dimension ''Student'', the base level ''Student'' and its association with the dimension. Lastly, the aggregated context ''Students by hours of study'' derives into its corresponding level and the association towards the previous level in the dimension (in this case ''student'' level). Table 2 summarizes the relationships between elements and its corresponding transformation rule.

Once the PIM has been derived, the analysts wish to change the requirements model. They wish to remove the ''Student'' context from the requirements. The context is removed from the model, and the previously aggregated context ''Students by hours of study'' now satisfies the information requirement from which ''Students'' derived in the CIM. With our approach we can track the changes done in the CIM model and identify the affected elements. In this case, the contexts associated with the information requirement in the CIM model will be affected, as well as the dimension ''DP_Students'' and its corresponding elements in the PIM model. The rule used to generate the PIM elements was Context2Dimension, so we will need to execute this rule again with the new parameters (in this case ''Students by hours of study'' context) to regenerate the affected part in the derived model. The result is shown in Fig. 11. In this model, the ''Students by hours of study'' dimension replaces the previous ''Students'' dimension, association and base level while the other elements remain the same.

This example is a simplification from a real-world project of another university. Three different data marts were designed, each data mart CIM averaging 40–50 decisional goals, deriving into an average of 28–34 contexts and measures per data mart. A change in a decisional goal would affect an average of other three to four goals and their derived contexts and measures, with the corresponding changes at the PIM level. Thanks to the traces, the designer knew which elements had to be changed in the final implementation, cutting the development time of the project.

6. Tool support

In this section we will present the implementation of our proposal, including the QVT rules specified in previous sections, in our tool Lucentia Business Intelligence Suite. The Lucentia BI Suite is a set of plugins developed for the Eclipse IDE, which allow us to model and develop DWs using a MDD approach.

According to our proposal for developing DWs [5,18], we follow a hybrid approach for DWs, starting from requirements specified in a CIM model. This CIM model is derived by means of automatic QVT transformations, presented in Section 4, resulting in a multidimensional PIM model of the DW as well as a trace model containing the traceability information. These QVT rules are implemented in our tool by using the ATLAS Transformation Language (ATL) [41], which has become increasingly popular for implementing model-to-model transformations, including those defined by means of the QVT standard.

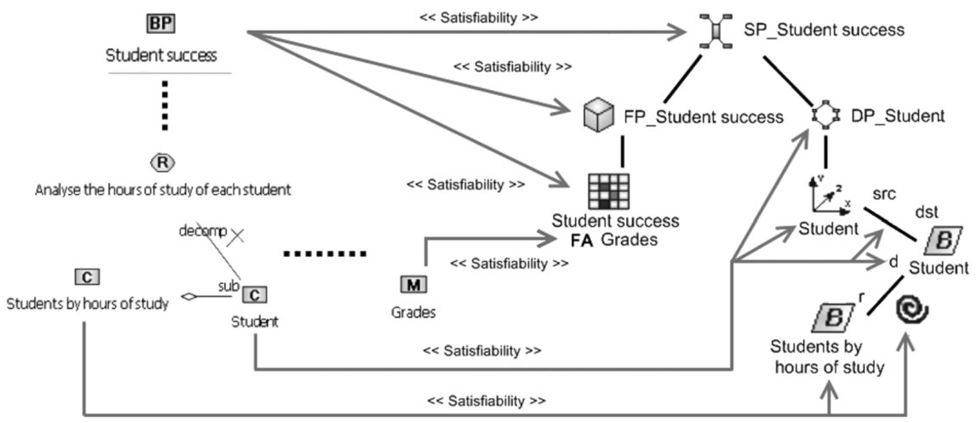


Fig. 10. Requirements modeled by using our i* profile for DW and its corresponding multidimensional model.

Table 2
Elements linked between CIM and PIM models by satisfiability links.

Rule name	Sources (CIM)	Targets (PIM)
BusinessProcess2Fact	Student success	SP_Student success, FP_Student success, Student success
Measure2FactAttribute	Grades	Grades
Context2Dimension	Student	DP_Student, Student(Dim) Student(Level), Association
Context2Base	Students by hours of study	Students by hours of study (Level), roll-up Association

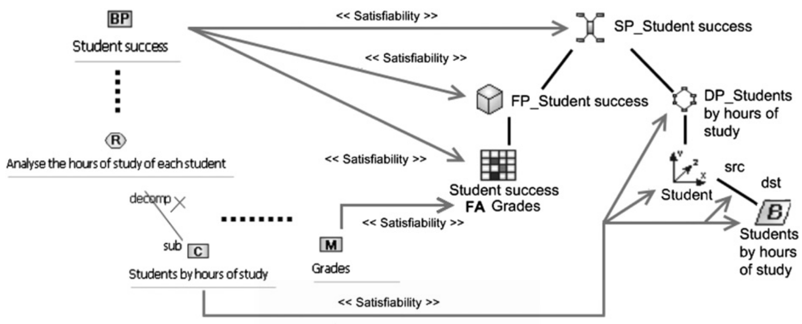
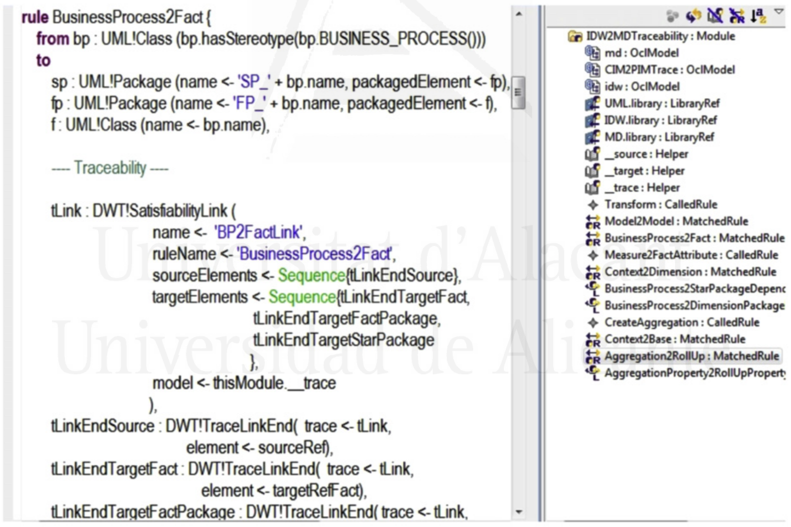


Fig. 11. New version of the CIM and its corresponding PIM obtained by applying our trace metamodel and the transformations.



```
rule BusinessProcess2Fact {
  from bp : UML!Class (bp.hasStereotype(bp.BUSINESS_PROCESS()))
  to
    sp : UML!Package (name <- 'SP_' + bp.name, packagedElement <- fp),
    fp : UML!Package (name <- 'FP_' + bp.name, packagedElement <- f),
    f : UML!Class (name <- bp.name),

    ---- Traceability ----

    tLink : DWT!SatisfiabilityLink (
                name <- 'BP2FactLink',
                ruleName <- 'BusinessProcess2Fact',
                sourceElements <- Sequence{tLinkEndSource},
                targetElements <- Sequence{tLinkEndTargetFact,
                                  tLinkEndTargetFactPackage,
                                  tLinkEndTargetStarPackage
                                  },
                model <- thisModule.__trace
            ),
    tLinkEndSource : DWT!TraceLinkEnd( trace <- tLink,
                    element <- sourceRef),
    tLinkEndTargetFact : DWT!TraceLinkEnd( trace <- tLink,
                    element <- targetRefFact),
    tLinkEndTargetFactPackage : DWT!TraceLinkEnd( trace <- tLink,
```

```
IDW2MDTraceability : Module
  md : OclModel
  CIM2PIMTrace : OclModel
  idw : OclModel
  UML.library : LibraryRef
  IDW.library : LibraryRef
  MD.library : LibraryRef
  __source : Helper
  __target : Helper
  __trace : Helper
  Transform : CalledRule
  Model2Model : MatchedRule
  BusinessProcess2Fact : MatchedRule
  Measure2FactAttribute : CalledRule
  Context2Dimension : MatchedRule
  BusinessProcess2StarPackageDepenc
  BusinessProcess2DimensionPackage
  CreateAggregation : CalledRule
  Context2Base : MatchedRule
  Aggregation2RollUp : MatchedRule
  AggregationProperty2RollUpPropert
```

Fig. 12. Implementation of QVT rules in ATL language inside the Lucentia BI tool.

In Fig. 12, we can see the implementation of the different QVT rules inside an ATL transformation. On the left side of the figure, we can see the ATL code for the "Business2Fact" rule defined. The "from" clause corresponds with the elements in the source metamodel, whereas the "to" clause corresponds with the elements in the target metamodels. Meanwhile, on the right side of the figure, we can see the different QVT rules implemented, as well as some auxiliary rules for managing the relationships between the fact and the different dimensions.

The reader may notice that, in our implementation, we only make use of the business process in the "from" clause of the ATL rule. ATL has a limitation in the declarative part, allowing each source element to be matched only once during the whole transformation. In turn, this would cause that only one measure could be derived according to the specified patterns. In order to overcome this limitation, the implementation considers each business process individually in the declarative part. This allows us to match each measure on its own, without requiring to match the business process multiple times.

Finally, each measure is related to its corresponding business process by means of imperative language, verifying the existence of the corresponding goal tree from the business process to the measure.

The execution of this transformation yields the results shown in Fig. 13. On the left side of the figure we can see the requirements model used in Section 5, which was used as the source of the transformation. From this requirements model, we obtained (i) the tree view of the trace model shown in the center part of the figure, and (ii) the target multidimensional PIM model, shown in the right side of the figure. As previously mentioned in Section 4, we can see that traces are structured in a hierarchical way, thus the satisfiability link between the measure and the fact attribute is stored inside the satisfiability link between the business process and the fact. In turn, this generates a resulting trace model which can be easily visualized, hiding the undesired details unless they are necessary.

Finally, once we have obtained the trace model, we can elaborate algorithms and visualization techniques which exploit the information stored in the traces. The algorithms can range from just simply analyzing the direct impact of modifying an element, to evaluating how many requirements are actually supported by the implementation of the DW, or even defining a policy for automatic propagation of changes, which establishes the actions to perform over the rest of DW models when a requirement or any other element is modified.

In this section we have presented the implementation of our proposal in the Lucentia BI tool. The Lucentia BI tool is a set of Eclipse plugins which allow us to model and automatically derive DW models. Now, we have extended the capabilities of our tool to easily trace user requirements to DW elements, thus being able to perform impact analysis and other interesting tasks over the models.

7. Conclusions and future work

In this paper, we have proposed a novel trace metamodel for DW development based on the MDA framework, in order to include semantic traces. We have shown the necessary trace models to be included in our development process. Furthermore, we have focused on the relationships between the CIM and the PIM and have proposed a set of QVT transformations to automatically generate the corresponding trace models. The great benefit of our proposal is the improvement in requirements validation and the identification of the corresponding elements in the PIM models, being able to easily assess the impact of changes and regenerate the affected parts. This has been shown by means of the presented example.

Our plans for the immediate future are developing a new set of QVT transformations to explore the relationships between the PIM and PSM and explore the potential of using the information recorded in the traces in order to support automated analysis. Furthermore, we are currently developing a traceability framework which takes care of the maintenance of traces and allows us to propagate changes from requirements to the PIM model in a fully automatic way using trace information. Finally, in the mid term and given the applicability of our proposal, we intend to analyze our approach using a rigorous analysis methodology and our implementation of an automatic change propagation algorithm, thus providing a clear estimation of the cost and efforts saved by using the fully automated approach.

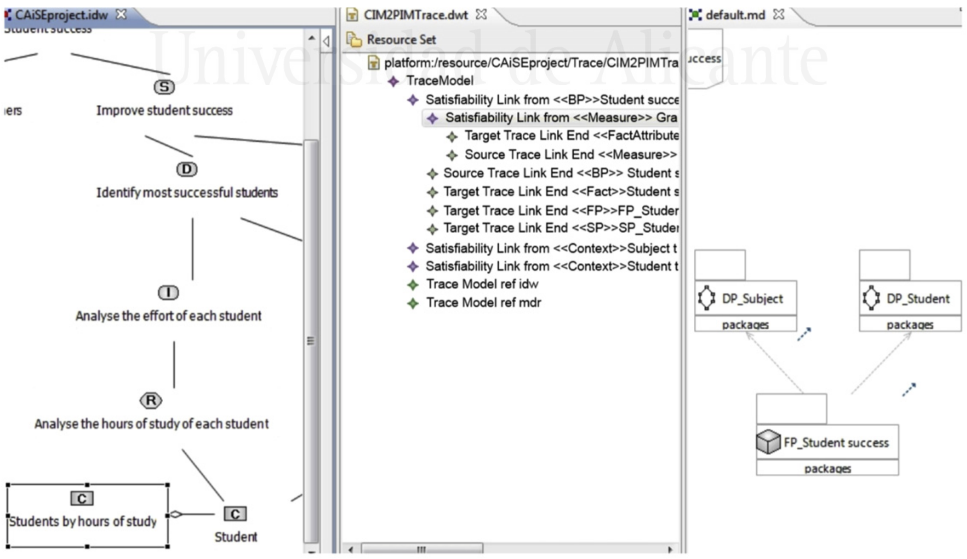

CAiSEproject.idw
Student success
S
Improve student success
D
Identify most successful students
I
Analyse the effort of each student
R
Analyse the hours of study of each student
C
Students by hours of study
C
Student
CIM2PIMTrace.dwt
Resource Set
platform:/resource/CAiSEproject/Trace/CIM2PIMTra
TraceModel
Satisfiability Link from <<BP>>Student succe
Satisfiability Link from <<Measure>> Gra
Target Trace Link End <<FactAttribute
Source Trace Link End <<Measure>>
Source Trace Link End <<BP>> Student s
Target Trace Link End <<Fact>>Student s
Target Trace Link End <<FP>>FP_Studer
Target Trace Link End <<SP>>SP_Studer
Satisfiability Link from <<Context>>Subject t
Satisfiability Link from <<Context>>Student t
Trace Model ref idw
Trace Model ref mdr
default.md
DP_Subject
packages
DP_Student
packages
FP_Student success
packages

Fig. 13. Generation of PIM and trace example models from requirements using the Lucentia BI tool.

Acknowledgments

This work has been partially supported by the MESOLAP (TIN2010-14860) and SERENIDAD (PEII-11-0327-7035) projects from the Spanish Ministry of Education and the Junta de Comunidades de Castilla La Mancha. Alejandro Maté is funded by the Generalitat Valenciana under the Grant ACIF/2010/298.

References

[1] W. Inmon, Building the Data Warehouse, Wiley-India, 2009.
[2] R. Kimball, The Data Warehouse Toolkit, Wiley-India, 2009.
[3] P. Giorgini, S. Rizzi, M. Garzetti, GRAnD: a goal-oriented approach to requirement analysis in data warehouses, Decision Support Systems 45 (1) (2008) 4–21.
[4] D. Ballou, G. Tayi, Enhancing data quality in data warehouse environments, Communications of the ACM 42 (1) (1999) 73–78.
[5] J.-N. Mazón, J. Trujillo, An MDA approach for the development of data warehouses, Decision Support Systems 45 (1) (2008) 41–58.
[6] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, The Data Warehouse Lifecycle Toolkit, Wiley, 2011.
[7] B. Ramesh, C. Stubbs, T. Powers, M. Edwards, Requirements traceability: theory and practice, Annals of Software Engineering 3 (1) (1997) 397–415.
[8] G. Spanoudakis, A. Zisman, Software traceability: a roadmap, Handbook of Software Engineering and Knowledge Engineering, 2005.
[9] S. Winkler, J. von Pilgrim, A survey of traceability in requirements engineering and model-driven development, Software and Systems Modeling 9 (2010) 529–565.
[10] N. Aizenbud-Reshef, B. Nolan, J. Rubin, Y. Shaham-Gafni, Model traceability, IBM Systems Journal 45 (3) (2006) 515–526.
[11] M. Barbero, M. Del Fabro, J. Bézivin, Traceability and provenance issues in global model management, in: ECMDA-TW, 2007, pp. 47–56.
[12] F. Jouault, Loosely coupled traceability for atl, in: ECMDA-TW, Nuremberg, Germany, 2005, pp. 29–37.
[13] R. Paige, G. Olsen, D. Kolovos, S. Zschaler, C. Power, Building model-driven engineering traceability classifications, in: ECMDA-TW, 2008, pp. 49–58.
[14] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, E. Merlo, Recovering traceability links between code and documentation, IEEE Transactions on Software Engineering 28 (10) (2002) 970–983.
[15] O. Gotel, A. Finkelstein, An analysis of the requirements traceability problem, in: Proceedings of the First International Conference on Requirements Engineering, ICRE, IEEE, 1994, pp. 94–101.
[16] O. Gotel, S. Morris, Macro-level traceability via media transformations, in: Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science, vol. 5025, Springer, Berlin, 2008, pp. 129–134.
[17] B. Ramesh, M. Jarke, Toward reference models for requirements traceability, IEEE Transactions on Software Engineering 27 (1) (2001) 58–93.
[18] J.-N. Mazón, J. Pardillo, J. Trujillo, A model-driven goal-oriented requirement engineering approach for data warehouses, in: J.-L. Hainaut, E. Rundensteiner, M. Kirchberg, M. Bertolotto, M. Brochhausen, Y.-P. Chen, S. Cherfi, M. Doerr, H. Han, S. Hartmann, J. Parsons, G. Poels, C. Rolland, J. Trujillo, E. Yu, E. Zimyie (Eds.), Advances in Conceptual Modeling Foundations and Applications, Lecture Notes in Computer Science, vol. 4802, Springer, Berlin, Heidelberg, 2007, pp. 255–264.
[19] P. Vassiliadis, Data Warehouse Modeling and Quality Issues, Ph.D. Thesis, Athens, 2000.
[20] S. Luján-Mora, J. Trujillo, I. Song, A UML profile for multidimensional modeling in data warehouses, Data & Knowledge Engineering 59 (3) (2006) 725–769.
[21] J. Mazón, J. Trujillo, J. Lechtenbörger, Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms, Data & Knowledge Engineering 63 (3) (2007) 725–751.
[22] OMG, A Proposal for an MDA Foundation Model.
[23] E.S.-K. Yu, Modelling Strategic Relationships for Process Reengineering, Ph.D. Thesis, Toronto, Ont., Canada, 1995.
[24] OMG, The Meta-Object Facility 2.0 Query/View/Transformation. Final Adopted Specification.
[25] A. Maté, J. Trujillo, A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses, in: H. Mouratidis, C. Rolland (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 6741, Springer, Berlin, Heidelberg, 2011, pp. 123–137.
[26] P. Arkley, P. Mason, S. Riddle, Position paper: enabling traceability, in: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, Edinburgh, Scotland, 2002, pp. 61–65.
[27] Y. Yu, J. Jurjens, J. Mylopoulos, Traceability for the maintenance of secure software, in: IEEE International Conference on Software Maintenance, 2008. ICSM 2008. IEEE, 2008, pp. 297–306.
[28] X. Franch, Incorporating modules into the i* framework, in: B. Pernici (Ed.), Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 6051, Springer, Berlin, Heidelberg, 2010, pp. 439–454.
[29] S. Walderhaug, E. Stav, U. Johansen, G. Olsen, Traceability in model-driven software development, Designing Software-Intensive Systems: Methods and Principle (2008) 133–159.
[30] M. Kolp, P. Giorgini, J. Mylopoulos, Organizational patterns for early requirements analysis, in: J. Eder, M. Missikoff (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 2681, Springer, Berlin, Heidelberg, 2003, pp. 617–633.
[31] H. Mouratidis, P. Giorgini, G. Manson, Integrating security and systems engineering: towards the modelling of secure information systems, in: J. Eder, M. Missikoff (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 2681, Springer, Berlin, Heidelberg, 2003, pp. 63–78.
[32] R.S. Kaabi, C. Souveyet, C. Rolland, Eliciting service composition in a goal driven manner, in: Proceedings of the 2nd International Conference on Service Oriented Computing, ICSOC '04, ACM, New York, NY, USA, 2004, pp. 308–315.
[33] Y. Yu, N. Niu, B. Gonzalez-Baixauli, W. Candillon, J. Mylopoulos, S. Easterbrook, J. do Leite, G. Vanwormhoudt, Tracing and validating goal aspects, in: Requirements Engineering Conference, 2007. RE '07. 15th IEEE International, 2007, pp. 53–56. http://dx.doi.org/10.1109/RE.2007.23.
[34] B. Amar, H. Leblanc, B. Coulette, A traceability engine dedicated to model transformation for software engineering, in: ECMDA-TW, 2008, pp. 08–12.
[35] M. Costa, A. Da Silva, RT-MDD framework—a practical approach, in: ECMDA-TW, Citeseer, 2007, pp. 17–26.
[36] A. Sousa, U. Kulesza, A. Rummler, N. Anquetil, R. Mitschke, A. Moreira, V. Amaral, J. Araújo, A model-driven traceability framework to software product line development, in: ECMDA-TW, 2008, pp. 97–109.
[37] B. Vanhooff, S. Van Baelen, W. Joosen, Y. Berbers, Traceability as input for model transformations, in: ECMDA-TW, Citeseer, 2007, pp. 37–46.
[38] S. Walderhaug, U. Johansen, E. Stav, J. Aagedal, Towards a generic solution for traceability in MDD, in: ECMDA-TW, Citeseer, 2006.
[39] M. Del Fabro, J. Bézivin, P. Valduriez, Weaving models with the eclipse AMW plugin, in: Eclipse Modeling Symposium, Eclipse Summit Europe 2006, Esslingen, Germany, 2006.
[40] H. Asuncion, A. Asuncion, R. Taylor, Software traceability with topic modeling, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, ACM, 2010, pp. 95–104.
[41] F. Jouault, I. Kurtev, Transforming models with atl, in: Satellite Events at the MoDELS 2005 Conference, Springer, 2006, pp. 128–138.

4

Traceability during the Reconciliation Process

The context of this chapter corresponds with the following papers:
Maté, A. *and Trujillo, J. Incorporating Traceability in Conceptual Models for Data Warehouses MDA. Proceedings of 30th International Conference on Conceptual Modeling (**ER'11**). 2011. Brussels, Belgium. Acceptance rate: 24.8%. ERA A*
Maté, A. *and Trujillo, J. and De Gregorio, E. and Song, Il-Y. Improving the Maintainability of Data Warehouse Designs: Modeling Relationships between Sources and User Concepts. Proceedings of 15th Workshop on Data Warehousing and OLAP, (**DOLAP 2012**). 2012. Maui, Hawaii. Acceptance rate: 30%. ERA B*

Data warehouses integrate several heterogeneous data sources by definition. Traditional data warehouse development approaches undergo a reconciliation process in order to identify if the data required is available and to find additional relevant data for decision makers. However, these techniques do not provide any means to document nor guide this step. Rather, they rely entirely on Extraction/Transformation/Load (ETL) processes. However, ETL processes (i) do not establish relationships at conceptual level using a multidimensional view, thus it is unclear what hierarchies and dimension levels are supported, (ii) are created after the implementation of the data warehouse, thus the information cannot be used to evaluate the data warehouse at design-time, (iii) do not preserve traceability to requirements, thus traceability in the process is lost, and (iii) establish detailed and complex steps in order to transform the data, divided into several files, whereas for the reconciliation process it is better to provide a simpler approach with better semantics that provides an overview of the relationships between the data warehouse and the data sources.

Given these problems, the second part of this PhD Thesis focuses on defining a modeling approach based on trace definition that enables the designer to model the relationships between data sources and the data warehouse, thus preserving traceability, simplifying the addition of a new data source, and providing an overview of the data integration process.

Incorporating Traceability in Conceptual Models for Data Warehouses by Using MDA

Alejandro Maté and Juan Trujillo

Lucentia Research Group
Department of Software and Computing Systems
University of Alicante
{amate,jtrujillo}@dlsi.ua.es

Abstract. The complexity of the Data Warehouse (DW) development process requires to follow a methodological approach in order to be successful. A widely accepted approach for this development is the hybrid one, in which requirements and data sources must be accommodated to a new DW model. The main problem is that the relationships between conceptual elements coming from requirements and those coming from data sources are lost in the process, since no traceability is explicitly specified, consuming additional time and resources. Previously, we have defined a trace metamodel in order to trace user requirements to DW conceptual models. In this paper, we complement our approach by including traceability along the successive refinements performed at the conceptual level. Therefore, we preserve the existing relationships between elements, eliminating additional costs derived from performing the matching process multiple times. We provide an example of how Query/View/Transformation rules can automate trace generation, and we also provide a set of guidelines for connecting conceptual elements coming from requirements with those coming from the data sources.

Keywords: Data warehouses, traceability, conceptual models, user requirements, data sources, MDA.

1 Introduction

Data Warehouses (DW) integrate several heterogeneous data sources in multidimensional structures (i.e. facts and dimensions) in support of the decision-making process [5]. Therefore, the development of the DW is a complex process which must be carefully planned in order to meet user needs. In order to develop the DW, three different approaches, similar to the existing ones in Software Engineering were proposed: bottom-up, top-down, and hybrid [3].

The first two approaches ignore at least one source of information for the DW, leading to failure in DW projects [3]. On the other hand, the third approach (hybrid) makes use of both data sources and user requirements [9], solving the incompatibilities by acommodating both requirements and data sources in a single conceptual model. Nevertheless, the acommodation process introduces modifications, causing the existing traceability by name matching to be lost. Once traceability is lost, the effort required for validating requirements or performing changes is increased, and the quality of the result is decreased [12]. The reason is that the developer must repeteadly track down each element through the different layers involved in the development process, which is time consuming and error prone. Despite this drawback, aside from our previous contribution in [7], where we defined a trace metamodel to trace DW requirements to their corresponding conceptual elements, the traceability aspect has been overlooked in DW development. By incorporating traceability, these time consuming and error prone tasks are minimized, allowing the developer to focus on the conceptual design of the DW, and improving the quality of the final product.

In this paper, we complement our previous works by including support for the traceability of conceptual elements through the different Platform Independent Models (PIM) up to the final conceptual model. We also provide an example of how trace generation can be automated where possible.

The remainder of the paper is structured as follows. Section 2 presents related work about traceability and DWs. Section 3 introduces the necessary trace semantics in order to include traceability at the conceptual level in DWs. Section 4 presents the QVT rules for automatic derivation of traces. Section 5 presents an example of application, in order to show the benefits of our proposal. Finally, Section 6 outlines the conclusions and further work to be done.

2 Related Work

In this section, we will briefly discuss the existing traceability research, its benefits and problems, and its current status in the DW field. Due to space constraints we will only describe the most important aspects.

Traditionally, traceability is focused on requirements. Either coming from the traditional RE [4,10] or following a MDD approach [1,2], requirements are traced to their lower abstraction level counterparts. Therefore, traceability helps assesing the impact of changes in requirements and rationale comprehension, by identifying which parts of the implementation belong to each requirement [2]. However, the effort required to manually record the traces, and the lack of standarization, make it difficult to apply traceability to projects. Therefore, there is a special interest on automating traces.

Our approach, presented in [9], applies MDD, and is sensitive to generate traces by exploiting transformation logic, thus being less error prone than manual recording. Therefore, by generating traces simultaneously as conceptual models are transformed, we provide support for requirements validation, impact change and automated analysis, while minimizing the drawbacks. While our approach applies MDA for DW development, other development proposals [3,11] make use of similar layers, so they could benefit from this approach.

In order to maintain all this information, elements coming from both requirements and data sources must be traced while maintaining the semantics of their relationships, allowing us to support automatic operations over the models.

3 Traceability from PIM to PIM DW Models

As previously stated, we require to trace information from both user requirements and data sources up to the final conceptual implementation. First, we will introduce the trace metamodel and the concepts used for tracing elements along the PIM models. Then we will describe how these elements will be traced.

In order to trace conceptual elements, up to the final PIM, we require to include different semantics, in order to differentiate the relationships between elements and support further automatic operations. These semantics are included in the trace metamodel (we refer the reader to [7] for more information) depicted in figure 1. The semantic types on which we will focus are:

– **Evolution** links are included to handle horizontal traceability which takes care of element changes at the same layer. In our case these links will track the different versions of each element at each PIM model.
– **Overlap** and **Conflict** are used for relating elements coming from both requirements and data sources in different shape. In this case, the developer will decide which is the correct solution to the conflict. These links are crucial for enabling traceability support, as they record the semantics between elements coming from data sources and those coming from requirements.
– **Rationalization** links are included as means of enabling the user to record his own annotations in the trace model about changes or decisions taken and provide reconciliated solutions for existing conflicts.

These trace types will be recorded in the different trace models included in our proposal, as shown in figure 2. In our proposal, first we derive an initial PIM

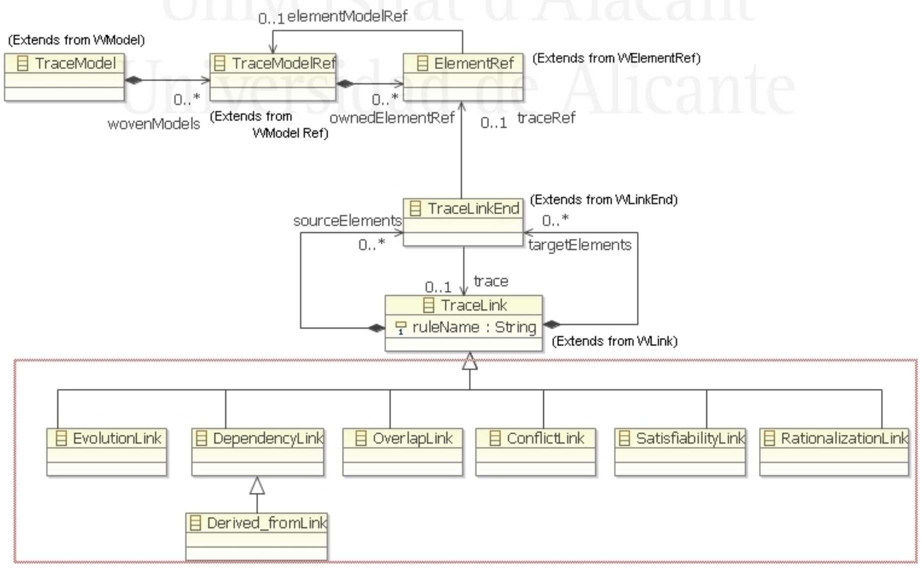


Fig. 1. Trace metamodel with semantic links for DWs

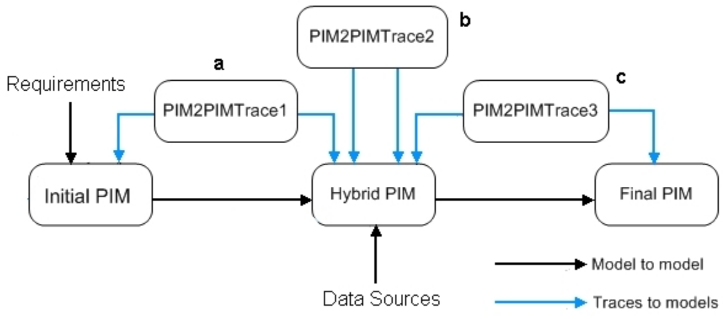


Fig. 2. Trace models linking the different PIM models in our DW approach

model from the user requirements represented in the requirements model. This PIM is refined with the necessary additions, not present at requirements level, and then it is derived into a mixed, hybrid PIM. The first trace model, labeled as "a" in figure 2, connects the initial PIM to the hybrid PIM in a pretty straightforward manner by means of *Evolution* traces. This trace model "a" is included in order to support automatic operations which require to track information related to requirements.

After we have derived the initial PIM, we first obtain a Platform Specific Model (PSM) from the data sources, which serves as basis to create a hybrid PIM model [8]. The hybrid PIM includes conceptual elements from both requirements and data sources and is characterized by representing the same concepts in different versions. In order to relate the different versions, their relationships are recorded by means of traces in trace model "b". These traces must be manually added because typically there is no knowledge about which element coming from the data sources is the counterpart to an element coming from user requirements. Therefore, we provide a set of guidelines in order to correctly relate elements in the hybrid PIM: whenever an element coming from requirements is complementary with its representation coming from data sources, they are related by means of *Overlap* links (G1). On the other hand, whenever an element coming from requirements is contrary to its representation coming from data sources, they are related by means of *Conflict* links (G2). In order to solve this situation, either one of the elements in conflict can be marked as solution, if it fits the user needs (G3) or, alternatively, the developer can provide a new, reconciliating element (G4), by means of *Rationalization* links.

Once the hybrid model has been refined, the desired elements which will be part of the final implementation are marked, as proposed in [9], and derived into the final PIM. Evolution traces, recorded in trace model "c" as part of the derivation into the final PIM, show which elements from the hybrid PIM were chosen to define the final conceptual model. This way, we can trace which parts of the final model come from either requirements or data sources, allowing us to perform impact change analysis as well as other automatic analysis tasks.

After having defined which trace models record the evolution of conceptual elements at PIM level, we will provide an example of how trace generation can be automated by means of transformations.

4 Automatic Derivation of Traceability Models in Data Warehouses

In this section, we will provide an example of how the necessary transformations can be formally defined to automatically generate the necessary traces. Due to paper constraints, we will only show one transformation rule as example.

According to our proposal for developing DWs [9], we use a hybrid approach, transforming models up to the final implementation by means of QVT rules. QVT rules specify a transformation by checking for a defined pattern in the source model. Once the pattern is found, a QVT rule transforms elements from the source metamodel into the target metamodel. In our case, a QVT which creates the *Evolution* link, from the hybrid to the final PIM, between overlapping bases in the hybrid PIM, is shown in figure 3.

In this QVT, two overlapping bases from the hybrid conceptual model, "b1" and "b2", are derived into a base "b3" in the final conceptual model.

On the left hand of the transformation rule, are the source metamodels. In our case, the sources are the multidimensional profile and the trace metamodel for DWs. On the upper left hand, we have a dimension "d1" and a base "b1", as well as the base level counterpart coming from the data sources, "b2". On the lower left hand, we have the traces which record the relationship between

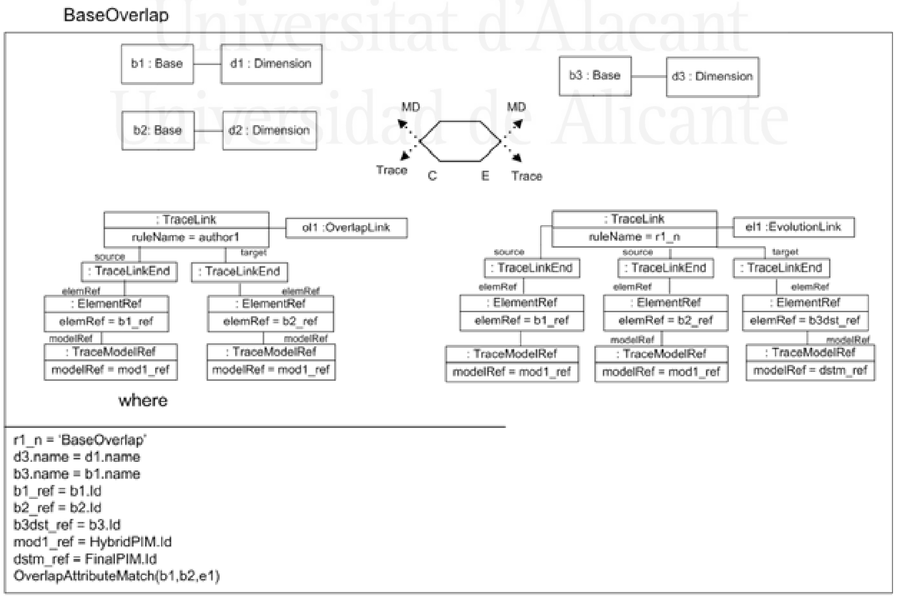


Fig. 3. QVT rule for deriving overlapping bases and creating their *Evolution* trace link

multidimensional elements coming from requirements, and those coming from the data sources. In this case, there is an overlap link between the two previously mentioned bases, which represents that both bases are complementary.

On the right hand of the transformation rule, are the target metamodels. On the upper right hand, we have our multidimensional profile, composed by the resulting dimension and base level. Since the relationship between the bases was defined as overlap, "b3" will present a combination of attributes from both "b1" and "b2". This merge will be performed by the *OverlapAttributeMatch* rule, called from the "Where" clause. On the lower right hand, we also have the trace metamodel, composed by the trace link which tracks the different elements used for composing the solution. In this case, as the original relationship between bases was an overlap, both bases are linked as sources of the new base level in the final PIM and its corresponding attributes.

The "C" at the center of the figure means that the source model is only checked, whereas the "E" means that the target models are enforced (generated). With QVT transformations, we can generate the associated traces simultaneously as the models are derived, avoiding the introduction of errors due to manual recording.

Once we have presented how to automate trace generation, we will present a case study for our proposal.

5 Case Study

In this section, we will present a case study for our proposal, showing how the traces can be used to relate the different elements in the hybrid PIM. This case study is inspired from a real world project with another university, and describes the basic process of our proposal, while making it easier to read the data source model. Note that the diagrams are presented with our iconography for DWs [6].

A university wishes to improve its educative process. In order to do so, a DW is designed to store the necessary information for the decision making process. The initial PIM, part of which can be seen at the left hand in figure 4, is derived from the users' requirements and refined with the expected attributes. This PIM includes 4 dimensions and a single measure. On the one hand, we have the *Subject* dimension. A subject is expected to include its code, a name, the credits and a description of the subject. Furthermore, subjects can be aggregated by their *Type*. On the other hand, is the *Teacher* dimension. A teacher includes a code, a name and the years of experience he has. Furthermore, teachers can be aggregated according to their *Department*, their *Faculty* or their job *Type*. The omitted dimensions in the figure, due to space constraints, are the *Student* and the *AcademicPeriod* dimensions.

As opposed to this initial PIM, the model created from the data sources (restricted to the most relevant tables) presents a higher number of attributes and lower readability. Part of this PIM can be seen at the right hand in figure 4. The first dimension is *TH_SUBJ*, which would correspond to the previous *Subject* dimension. This dimension includes a code for the subject, as expected,

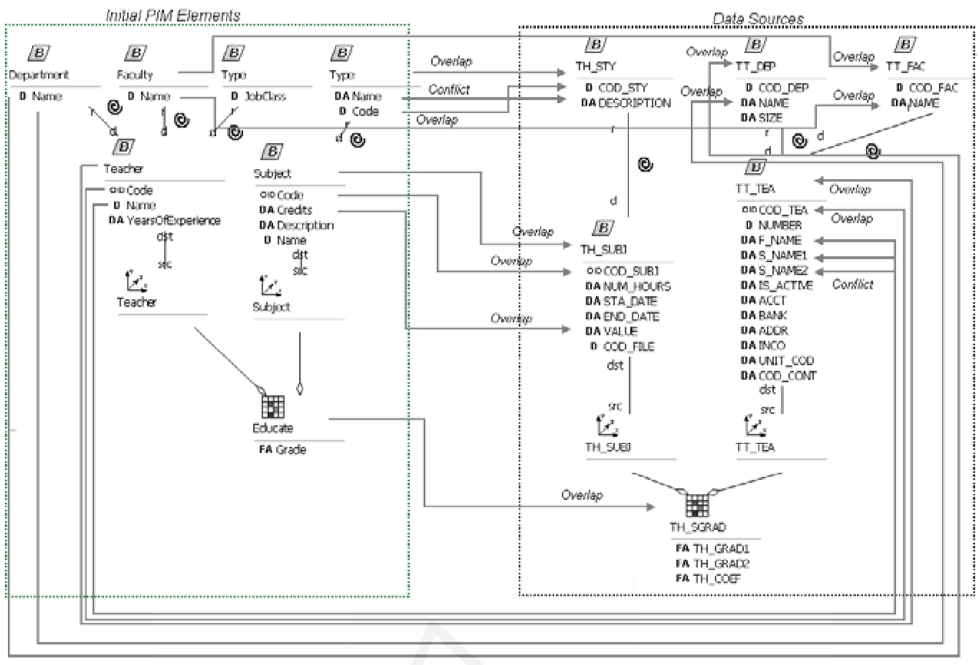


Fig. 4. Intra-model PIM traces relating conceptual elements from requirements (left) with elements from data sources (right)

the number of hours of the subject, a starting date, an ending date, a value which could correspond to the number of credits, and a code for the file of the subject. Subjects may also be grouped by type, as expected. The next dimension is *TT_TEA*, corresponding to information about the teachers. The information recorded for a teacher includes his name and surname, a mark for indicating if he is active or not, his bank information, address, unit code and a code related to the accounting. According to the data sources, teachers can be grouped either by department or by faculty. In this case, if we wished to group them by their job position, additional elements would be required.

Once we have both models in the hybrid PIM diagram, we can manually record the traces relating their elements, as sketched in figure 4. By recording only once these relationships, we do not require to repeteadly match each element coming from the requirements with those in the data sources, avoiding the introduction of errors in the process.

6 Conclusions and Future Work

In this paper, we have proposed a traceability approach in order to explicitly specify the relationships between elements at the conceptual level in DWs. We have shown the necessary trace semantics to record these relationships and have proposed a set of guidelines, in order to aid with the identification of these relationships. Furthermore, we have shown how trace derivation and recording would be automated and have exemplified the application of the proposal by means of the case study. The great benefit of our proposal is that the reconciliation task is only performed once per element and is preserved for further derivations. Therefore, we avoid repeteadly inspecting the data sources in order to match conceptual elements coming from requirements with those coming from data sources, diminishing time and resources spent.

Our plans for the immediate future are defining the complete set of QVT transformations to derive alternative final PIM models and to explore the relationships between the PSM and PIM levels.

Acknowledgments. This work has been partially supported by the MESOLAP (TIN2010-14860) and SERENIDAD (PEII-11-0327-7035) projects from the Spanish Ministry of Education and the Junta de Comunidades de Castilla La Mancha respectively. Alejandro Maté is funded by the Generalitat Valenciana under an ACIF grant (ACIF/2010/298).

References

1. Aizenbud-Reshef, N., Nolan, B., Rubin, J., Shaham-Gafni, Y.: Model traceability. IBM Systems Journal 45(3), 515–526 (2006)
2. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. IEEE Transactions on Software Engineering 28(10), 970–983 (2002)
3. Giorgini, P., Rizzi, S., Garzetti, M.: GRAnD: A goal-oriented approach to requirement analysis in data warehouses. DSS 45(1), 4–21 (2008)
4. Gotel, O.C.Z., Morris, S.J.: Macro-level Traceability Via Media Transformations. In: Rolland, C. (ed.) REFSQ 2008. LNCS, vol. 5025, pp. 129–134. Springer, Heidelberg (2008)
5. Kimball, R.: The data warehouse toolkit. Wiley-India (2009)
6. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. DKE 59(3), 725–769 (2006)
7. Maté, A., Trujillo, J.: A Trace Metamodel Proposal Based on the Model Driven Architecture Framework for the Traceability of User Requirements in Data Warehouses. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 123–137. Springer, Heidelberg (2011)
8. Mazón, J., Trujillo, J.: A model driven modernization approach for automatically deriving multidimensional models in data warehouses. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 56–71. Springer, Heidelberg (2007)
9. Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. DSS 45(1), 41–58 (2008)
10. Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Transactions on Software Engineering 27(1), 58–93 (2001)
11. Vassiliadis, P.: Data Warehouse Modeling and Quality Issues. Ph.D. thesis, Athens (2000)
12. Winkler, S., von Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Software and Systems Modeling 9, 529–565 (2010)

Improving the Maintainability of Data Warehouse Designs: Modeling Relationships between Sources and User Concepts

Alejandro Maté
Lucentia Research Group
University of Alicante
Alicante, Spain
amate@dlsi.ua.es

Juan Trujillo
Lucentia Research Group
University of Alicante
Alicante, Spain
jtrujillo@dlsi.ua.es

Elisa de Gregorio
Lucentia Research Group
University of Alicante
Alicante, Spain
edg12@dlsi.ua.es

Il-Yeol Song
College of Information Science and Technology
Drexel University
USA
song@drexel.edu

ABSTRACT

In data warehouse (DW) development, a series of mappings must be specified between user concepts and data source elements, in order to identify which sources must undergo an integration process. Until now, these mappings are either assumed to be implied by name matching or identified according to the designer's experience. Then, the result is implemented as Extraction/Transformation/Loading (ETL) processes. Since ETL processes relate elements at the logical level, designers cannot adequately analyze how a change in requirements or in the data sources affects the analysis capabilities. Furthermore, this approach makes it difficult to perform incremental changes in DW design, requiring in some cases to perform the whole analysis again. In this paper we present a set of semantic mappings that relate user concepts specified by requirements to those obtained from data sources. In turn, this allows us to accurately identify how any potential change affects the different structures and ETL processes. As a DW evolves over time, our approach easily allows us to incorporate new concepts, as well as any change introduced at requirements or data sources into the DW repository with no need to redesign the whole DW. In order to show the application of our proposal, we show a real case study focusing on the Digital library of the University of Alicante.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database administration—*Data warehouse and repository*

General Terms

Design, Theory

Keywords

Data warehouses, reconciliation, evolution, maintainability.

1. INTRODUCTION

Data warehouse (DW) development requires to extract information from multiple heterogeneus sources in order to support the decision making process [9]. In DW development, user requirements specify the information needs [7, 14] that must be met and determine the structure of the target DW. Once the structure has been defined, the DW is populated with information coming from different data sources, which are exploited during decision making process.

Nevertheless, the naming conventions and structures of data sources may not always match the structure specified by user requirements since (i) they are designed with different objectives than Online-Analytical Processing (OLAP), and (ii) the target structure is the (expected) result of integrating the different data sources each following its own conventions [3]. Therefore, in order to evaluate the viability of the DW, a series of mappings must be identified between user concepts and data source elements. After these mappings have been identified and the DW has been implemented, the integration process is implemented as Extraction/Transformation/Loading (ETL) processes in their corresponding files [4].

However, ETL processes only capture the results of these mappings at the logical level, while mappings themselves are not recorded in any diagram. In turn, analyzing ETL processes alone ignores key aspects related to the analysis needs. For example, consider Figure 1. An ETL process extracts information from "Document_TD" table and loads it into "Publication" and "Time" DW tables. Compare the information provided in this ETL flow with the one captured in Figure 2. In this Figure, a conceptual MD model of the data sources shows column "publicationMentio" related with user concepts by means of a Solvable Conflict relationship

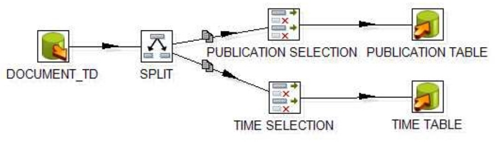


Figure 1: A single data source column determines the viability of multiple levels. ETL view

(SC, see Section 3.2)). This Figure shows that (i) this single data source column "publicationMentio" actually provides the necessary information to identify three different analysis levels "Place", "Province" and "Time", and (ii) "Country" information is missing thus we require to gather it from another source. Therefore, if any change is performed on the source column or if additional information is provided, the designer can quickly identify the impact of these changes.

As shown, by following a traditional approach, designers cannot adequately document the correspondences between user concepts and data sources. In turn, designers cannot accurately assess how a change in the concepts required by decision makers or in the data sources will affect the DW. In some cases, changes may require to exhaustively analyze all the documentation available, such as when new data is available and some requirements had not been previously met. Thus, with current approaches, the maintainability of the whole system is impacted negatively [4].

In our previous work [11, 12, 13, 14, 16], we defined a hybrid DW development approach in the context of the Model Driven Architecture (MDA) [10]. In our approach, requirements are specified in a Computation Independent Model

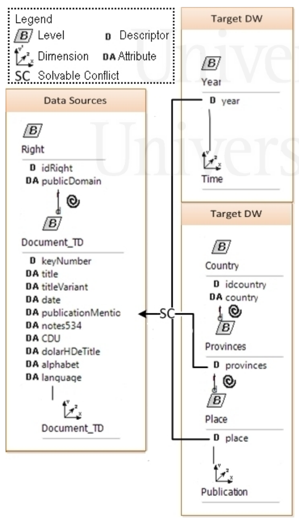


Figure 2: A single source column determines the viability of multiple levels. Conceptual view

(CIM) by means of a goal-based UML profile [14] extending the i* framework [21]. Then, they are automatically derived into a conceptual DW model [11], and reconciliated with the data sources by using reverse engineering [15].

In this paper, we present a set of semantic mappings specifying the existing relationships between user concepts specified by requirements and concepts derived from data sources. Our proposal allows us to (i) identify which DW structures and ETL processes will be affected by a change either in user concepts or in the data sources, (ii) identify which user requirements can be satisfied, as well as calculate different quality metrics over them [19], and (iii) preserve the mapping between the target DW and the data sources, thus allowing us to instantiate the mappings in the form of conceptual ETL processes [6, 17].

The remainder of this paper is structured as follows, Section 2 briefly presents the Related Work in ETL processes, DWs, and data provenance. Section 3 describes the formalization of the relationships between conceptual DW elements and data source elements. Section 4 describes the application of the proposal to a case study from the digital library of the University of Alicante. Finally, Section 5 summarizes the conclusions and sketches the future work.

2. RELATED WORK

The matching and reconciliation process between requirements and data sources has been tackled in different DW design approaches [7, 8, 15]. Our research shows that we can differentiate two basic aspects in this process: how the matching is performed and how DW structures can be traced to their original counterparts.

On one hand, in order to perform the matching, hybrid DW approaches such as [1, 7, 15] first obtain a reverse engineering model of the data sources. Then, a name matching is performed, fusing together structures derived from requirements and those obtained from the data sources. However, as described in [5], the language employed by decision makers to describe DW requirements is different than the one used by engineers to design an OLTP schema or other sources such as plain text. Therefore, the name matching process may not obtain the expected result. Furthermore, even if a partial solution such as previously defining a common terminology is applied [1, 18], existing structural differences still require to perform a manual reconciliation process. In addition, each time a new data source is added, this process will have to be repeated in order to integrate new and previously existing data.

On the other hand, another important aspect is how structures and data are traced to their original counterparts. As both data sources and requirements evolve over time, this evolution influences the DW. Data sources, such as OLTP databases, are updated as business processes change. In turn, at least some of this changes must be propagated to the DW supporting the organization's Business Intelligence system in order to provide correct data and being able to take adequate decisions. In a similar way, requirements also evolve as the business strategy adapts to constant changes in the environment, such as new competitors appearing, sales decreasing due to an economic crisis, etc. This leads decision makers to require additional information that may not have been previously considered, thus requiring to modify not only the target DW but also the ETL processes.

Traditionally, this aspect is tackled by means of traceability [20] and data provenance [2] approaches. On one hand, traceability is focused on tracking the relationships between different elements involved in the design process as they evolve through a set of operations. However, most DW modeling approaches rely on ETL processes to trace DW structures and data lineage back to their origins. As ETL processes do not model DW structures at the conceptual level, traceability between the multidimensional concepts and the data sources is lost. In addition, user concepts which were not incoporated in the target DW are removed from the final diagram in the reconciliation process. Therefore, when incorporating new data they may be overlooked, thus losing analysis capabilities. On the other hand, data provenance is focused on identifying where the data stored came from and which operations were performed to obtain its current value. However, implicit data provenace provides no mechanisms to record the mappings and transformations performed to user concepts as a result of the reconciliation process. As a result, implicit data provenance serves to trace back to its origins the data used but cannot be applied to user concepts as they include no data.

Summarizing, current DW development approaches do not provide any mechanism to record the mappings involved in the reconciliation process, thus decreasing the maintainability of the system. This process is typically approached by name matching, despite the different name conventions employed in requirements and data sources. Moreover, any structural differences are to be solved by the designer using his own methods. Finally, analysis of the impact of changes cannot be adequately performed since (i) there is no documentation specified recording the structural changes, and (ii) ETL processes relate sources and DW structures at the logical level. Our approach solves these drawbacks by providing a set of concepts to model these mappings. In turn, traceability towards requirements and data sources is also preserved as described in [13]. In this way, we can easily identify which elements should be integrated when new data sources are added or previously existing data sources and requirements are updated.

3. RELATIONSHIPS BETWEEN USER CONCEPTS AND DATA SOURCES

In order to address the maintainability problem in DW design, we propose to model the relationships established in the reconciliation process explictly, by comparing user expectations with the information provided by the data sources. This comparison is performed by relating the conceptual DW model obtained from requirements [11] with the different conceptual models obtained from data sources by means of reverse engineering [15], which capture real data characteristics and structure.

3.1 Overview of the Approach

In our approach, we make explicit the implicit knowledge in the reconciliation process, avoiding information losses and improving the maintainability of the system. This is performed by introducing a set of semantic mappings. In our development process, shown in Figure 3, we start by obtaining a conceptual model of the DW from user requirements (1). Afterwards, by means of reverse engineering, we obtain a multidimensional model of the data sources (2). Once we have both models, we apply our proposal to capture the semantic relationships between them (3), and finally derive the target DW according to the relationships modeled.

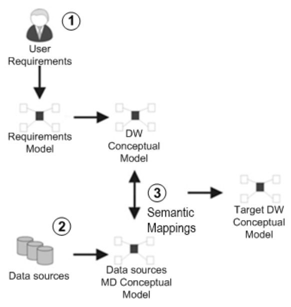


Figure 3: Steps in the development process

In the following, we focus on the definition of these relationships in order improve the maintainability of the DW. Then, we show the applicability by means of a case study.

3.2 Formal Definition

In order to adequately relate elements and analyze how a change affects the DW, we must be able to capture the semantic of the relationship between user concepts and data. In this section, we present the formal definitions involved in our approach. First, we start by defining the basic elements. Then, we define the different relationships which can be identified according to the set theory by analyzing how user expectations compare to data provided.

Conceptual DW models are defined by means of multidimensional modeling, specifying sets of Facts (center of the analysis) and Dimensions (context of analysis). According to the definitions from [16] D is a set of dimension schemata such that two distinct dimension schemata $D_1, D_2 \in D$ only have dimension level l_{All}. Each Dimension Schema is a quadruple $D = (N, L, \preceq, C)$ where N is the dimension name, L is a finite set of dimension levels, $\preceq$ is a lattice, specifying a dimension hierarchy, such that (a) $infL$ is not null, and (b) $supL = l_{All}$; where $infL$ represents the root level and $supL$ represents the highest level; and C is a (possibly empty) set of context dependencies.

In addition to these definitions, we formally define Levels and Attributes. A Level L_i is a pair $L = (N, A)$ such that N is the name of the level, and A is a set of attributes over this level. One of these attributes is the identifier of the level and satisfies $Des(a_i)$. Finally, an attribute a_i is an atomic element, which takes values from a given domain D_a.

After providing the formal definitions for the different elements composing a multidimensional schema, we focus on specifying the correspondences between user concepts obtained from requirements and conceptual elements obtained from data sources. On one hand, user concepts define the space of possible values for each element by describing their characteristics. As such, domains corresponding to requirements will be defined by intension, i.e. $D_1 = \{$"all provinces in the country"$\}$. On the other hand, data sources are characterized by providing specific sets of instances, thus domains corresponding to data sources will be defined by extension, i.e. $D_2 = \{$"Alicante","Valencia","Castellón",...$\}$.

Analyzing the existing similarities and differences between both sets will allow us to categorize their relationships and provide accurate information which can be used by the designer when incorporating changes.

3.2.1 Attribute Analysis

Now we proceed to analyze the different kinds of relationships that may be established between conceptual elements. We start by analyzing the most basic element: attributes. Attributes can be compared by analyzing their domains. The first aspect to consider is if the domains of the different attributes being related are the same or not. For example, consider the code for a document in the Digital Library. The decision maker specifies that each document has a code, expected to be composed by a sequence of numbers. Indeed this code is a sequence. In this case both domains would be the same, thus no transformation would be necessary to go from one domain to the other. We categorize this situation as an *Overlap* (O). However, this is not always the case. Consider the language of each document. The decision maker expects to analyze a set of language names, such as "English", "French", etc. Instead, the data sources store languages recording their language code. While both attributes refer to languages, codes cannot be used directly in the DW and require to be transformed in order to obtain the expected language names. This difference between the domains of both attributes is categorized as a *Conflict* (C).

$$\left.\begin{array}{l} a_1 = \mathit{idDocument} = \{\text{document codes}\} \\ a_2 = \mathit{keyNumber} = \{11111, 22222, ...\} \end{array}\right\} \Longrightarrow \text{O}(a_1,a_2)$$

$$\left.\begin{array}{l} a_3 = \mathit{name} = \{\text{names of languages}\} \\ a_4 = \mathit{language} = \{aar, abk, afk, ...\} \end{array}\right\} \Longrightarrow \text{C}(a_3,a_4)$$

By performing a thorough analysis w.r.t. the set theory we can differentiate six different categories. These categories allow the designer to identify mismatches between the information expected and the information provided:

1. An *Equivalent Overlap* (EO) relationship holds between two attributes when both domains define the same set. For example, we have "idDocument" and "keyNumber" as previously shown.

2. A *Subset Overlap* (UO) relationship holds between two attributes when the second attribute has a restricted domain compared to the first one. Thus, the second domain is strictly included in the first one. For example, if we expected "idDocument" to include the identifiers of documents from all the different libraries being integrated, but instead "keyNumber" did not have the identifiers of one or more libraries available.

3. A *Superset Overlap* (SO) relationship holds between two attributes when the second attribute has a more general domain compared to the first one. Thus, the second domain strictly includes the first one.

4. A *Complementary Overlap* (CO) relationship holds between two attributes when both domains share a common part but none of them is fully included in the other. For example, if we expected "type" in "Type" level to include "handwritten" and "digital" formats and, instead, it includes "handwritten", "music composition", and "theater".

5. A *Solvable Conflict* (SC) relationship holds between two attributes when a function F is necessary to project elements from the second domain into the first one, and such function exists. For example, "name" expecting a language name and instead we have "language" codes, from which we can obtain the expected set of names.

6. An *Irresolvable Conflict* (IC) relationship holds between two attributes when a function F is necessary but this function does not exist. For example, if we expected the "idDocument" to be the name of the document and we had "keyNumber" as the only element provided which stores only sequences of numbers.

7. Finally, *No Relationship* (NR) is the absence of a correspondence under any of the previously-defined relationships, such as when two attributes do not have anything in common.

A summary of the previous relationships is shown in Figure 4 in the form of trace metaclasses, where the four Overlap relationships described are a specialization of the Overlap concept, while the two Conflict relationships described are a specialization of the Conflict concept from [12]. All the relationships can have n source target elements.

We specify subset and superset relationships by means of proper inclusion relationships between sets. This is because we wish to identify the cases when there is less information than required and when there is more information than required. For example, imagine that the decision maker wishes to include only english authors in the DW, such that D_1=$\{$*ids of English authors*$\}$. However, the data source we are using include information from all the authors available (SO), thus we have additional authors. This data may be useful information, or, alternatively, it can turn into noise which should be filtered out. Conversely, the lack of information (UO) can derive into erroneous decisions, which are counterproductive for the organizations. For example, considering an author is not important because he created a few documents while lacking information of certain books. These scenarios cannot be detected by means of constraints over the summarizability of measures, but they can be analyzed thanks to the categorization proposed.

3.2.2 Level Analysis

Once we have defined the different kinds of relationships between attributes, which are considered atomic, we can proceed to define the relationships between coarser-grain elements in the multidimensional model. The second element we analyze is levels. Levels, $L = (N, A)$, are combinations of attributes (A) paired with a semantic name (N), giving the whole set of attributes a meaning. The most important attribute in a level is its descriptor, $Des_{L1}(a_1)$, since it identifies the instances of that level. For example, compare a "Document" level (Figure 5), expected to be identified by an id, with a "Document_TD" level from the data sources (Figure 6), also identified by an id. If both identifiers are sequence of numbers, they will be overlapping with each other, thus the DW will be able to provide each instance of "Document" as was expected. As such, both levels would be overlapping, and the amount of information provided by each one would depend on the rest of attributes. However, if both levels used different types of identifiers, i.e. expecting to identify a "Document" by its title, then we would obtain

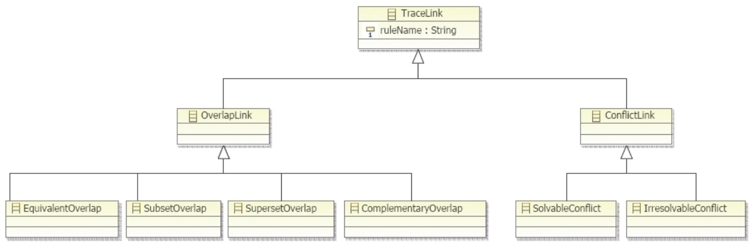


Figure 4: Categorization of Overlap and Conflict relationships between requirements and data sources

a different result than we expected since a title can be repeated for multiple codes. Therefore, both levels would be conflicting and would require a transformation in order to obtain the correct data.

We now present the definitions used to relate levels, allowing the designer to identify mismatches between how concepts are identified, and how much information is provided:

1. An *Equivalent Overlap* (EO) relationship between two levels holds when both levels have all their attributes are related such as in the case of "Author". Furthermore, in order to guarantee the compatibility, an overlap between descriptor attributes, $O(Des(l_1), Des(l_2))$, must hold in every overlap relationship. For example, "Author" (data source) presents exactly the expected set of attributes in "Author" (requirements), thus they have an EO relationship.

2. A *Subset Overlap* (UO) relationship between two levels holds when the second level provides less information than the first one, thus its set of attributes is a subset of l_1. For example, if we had a "Country" (data source) level providing only the descriptor.

3. A *Complementary Overlap* (CO) relationship between two levels holds when both levels share a set of common attributes in addition to other attributes that are exclusive to each of them. For example, "Document_TD" is missing the expected "uuid" attribute, but instead provides additional information such as "CDU", "Notes534", or "Date".

4. A *Solvable Conflict* (SC) relationship between two levels holds when their descriptors are conflicting or they are not related to each other but, instead, are related to other dimension attributes. Therefore, the way of identifying each level is different. For example, we can obtain "Alphabet" from "Document_TD" by extracting the data from the "alphabet" column. However, these two levels have a different meaning (identifier), thus we are actually transforming one into the other.

5. An *Irresolvable Conflict* (IC) relationship between two levels holds when their descriptors are not related or they present an IC between them, thus one level cannot be converted into the other. For example, "Language" cannot be obtained from "Document_TD" by means of a transformation since we are missing the required descriptor for the level.

Complementary Overlaps are common when the user requires to analyze information from concepts which are entities in the data sources. Usually, the identifier employed is the same. However, it is rare that all the required attributes are stored directly in the data source nor every attribute stored is explicitly enumerated in the requirements.

Conflicts between levels involve a mismatch between the semantics of levels, and they are common when the user requires to analyze in detail concepts which are not entities on their own in the data sources. This situation can be seen in Figures 2 and 7 where we require to extract the "Year", the "Place" of publication of a given document, and the "Language" of the document.

3.2.3 Dimension Analysis

The last element we analyze are dimensions. A dimension is a quadruple $D = (N, L, \preceq, C)$. Dimensions are named (N) sets of levels (L) which satisfy a partial order relationship $(\preceq)$ which may depend on context dependencies (C). For example, in Figure 5, dimension "Document" includes a base level "Document" which is previous to every other level. On the other hand, "SupportForm" and "Volume" cannot be ordered as they are aggregated in parallel paths.

Typically, the name of the dimension is the same as the name of the lowest level in the Lattice, since this level identifies the tuples of the dimension in a similar way as *Descriptor* attributes for levels. Thus the lowest level is one of the critical elements when comparing two dimensions. If the lowest level is overlapping or its conflict can be solved, then, the relationship between dimensions will depend on how their partial order relationships behave. However, if the lowest level is not related or presents no solution, then, it will not be possible to obtain the required dimension. For example, in Figure 7, the expected "Language" level descriptor could not be related with any other element. Therefore, there is no way to obtain the basic set of instances for the "Language" dimension.

Additionally, in the case of dimensions, the Lattice constitutes another critical element, since it defines the possible aggregation paths. If the expected $\preceq$ relationship is altered, and two levels interchange their order, then, both dimensions will be conflicting. Altering the partial order implies that the actual aggregation paths are contrary to what was expected, thus a transformation will be required in order to provide the order expected by the user. For example, an advanced case is shown in Figure 8, where "Format" corresponds with "SupportForm". However, "Format" is not aggregated by any level, thus a transformation will be required in order to combine "Document_TD" instances with their corresponding "Format", thus obtaining the expected "Document" dimension. In the following we present the categorization used to relate dimensions, allowing the designer to identify structural differences in the dimensions between the expected analysis hierarchies and the data retrieved:

1. *Equivalent Overlap* (EO) relationship holds between two dimensions when both dimension present the same set of levels as well as the same Lattice. As with levels, in every overlap relationship it is required that the lowest level is related by either an Overlap or Solvable Conflict relationship, in order to guarantee that we can analyze the data at the finest aggregation level specified by the dimension. For example, the "Author" (data source) dimension provides exactly the expected hierarchy of levels.

2. *Subset Overlap* (UO) relationship holds between two dimensions when the expected dimension presents more levels or relationships while maintaining the partial order between levels related by Conflict or Overlap relationships. For example, if we expected a "User" dimension representing the users of the Digital Library and instead of two levels "User" and "Category", data sources provided only "User" level.

3. *Complementary Overlap* (CO) relationship holds between two dimensions when each dimension presents levels or relationships that the other dimension does not include, while also maintaining the partial order between levels related by Conflict or Overlap relationships. For example, "Document_TD" dimension includes the "Right" level, which was not expected, but lacks the "SupportForm" and "Type" levels.

4. *Solvable Conflict* (SC) relationship holds between two dimensions when the partial order relationship is altered. In other words, levels in each dimension Lattice are crossed in the other Lattice. For example, "Format" dimension presents the "Format" level as the first level, instead of aggregated from individual documents as expected in "Document".

5. *Irresolvable Conflict* (IC) relationship holds between two dimensions when the lowest level in the first dimension is not related or presents an IC with the other level. Therefore, we cannot obtain a transformation to relate the first dimension with its corresponding fact. The solution for an IC is to look for a common higher aggregation level. For example, we cannot obtain the required instances of the "Language" dimension since the lowest level is lacking its identifier but we can correctly obtain "Alphabet" instances.

4. CASE STUDY

In this section we will present the application of our conceptualization to a real case study: the integration process in the Digital library of the University of Alicante.

Recently, the digital library in the University of Alicante performed an integration process by combining different data sources. In order to analyze the information related to documents, the Digital library included sources modelled according to different standards for digital works, such as FOAF or

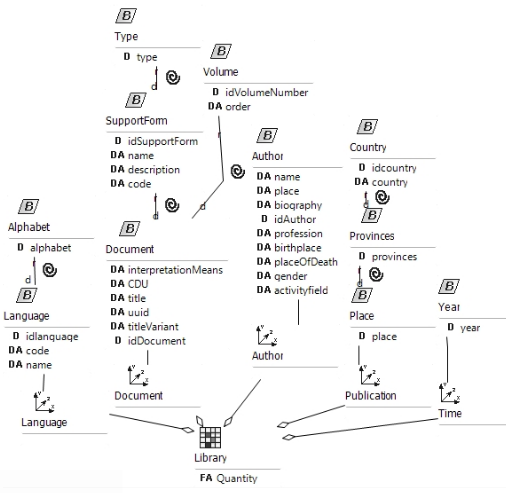


Figure 5: Conceptual model satisfying user requirements for the DW

MARC21 ontologies among others. Each of these standards specifies its own way to name and structure information. As such, certain fields contain multiple information, such as the title, the name of the author and the date published, together in a single field, separated by special characters ("$"). Furthermore, some of this information was optional, and did not always appear in the same field. Moreover, tracking all this information was a complex task, since there was duplicated information present, such as variations including alternative titles in different columns or multiple indexes relating the same concept.

Although each source was designed according to a standard, the structural differences and naming conventions made impossible to successfully apply a name matching approach. Therefore, in order to analyze which requirements could be satisfied, identify critical attributes, and analyze the impact of changes, we applied our approach to relate the data sources with the expected model in one data mart.

Our approach starts by obtaining the expected DW model from user requirements [13]. The resulting model is shown in Figure 5. In this model, we analyze the digital documents in the library. In this model, the most important dimension is "Document", which represents the different documents in the library. From each document we need to know its "title", the Universal Decimal Classification ("CDU") which classifies the document and the universal unique identifier ("uuid"). This dimension has three additional aggregation levels: the format it is stored ("SupportForm"), the "Type" of document, and additionally its "Volume". There are also other dimensions involved in the analysis: the "Author" of the document, its "Language", where it was published ("Publication"), and the date when it was published ("Time").

Once the requirements model has been obtained, the next step is to obtain a multidimensional model from the sources. This step can be performed either manually or by means of

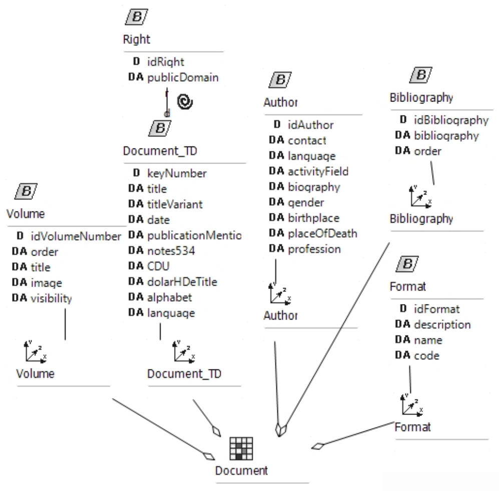


Figure 6: Conceptual model obtained from data sources

algorithms which apply some of the existing heuristics to obtain a multidimensional model from relational sources [16]. For the sake of simplicity, we only show a handful attributes in the conceptual model although some tables presented over 40 attributes. The result of the modeling step is shown in Figure 6. In this figure we can observe some differences w.r.t. the requirements model previously described. Some dimensions are initially missing like "Publication", "Time", and "Language", while others, such as "Bibliography" appear. Moreover, some requirements levels like "Volume" and "SupportForm" appear as dimensions "Volume", "Format". Finally, some attributes are missing, such as "activityField" in "Author", while other new ones appear, like "notes534".

However, these differences do not mean that the necessary data do not exists in the data sources. Instead, important structural differences must also be considered. In order to capture this differences and be able to analyze the correspondences between requirements and sources, we apply our proposal to the mapping between both schemata[1]. By applying our approach, we are able to identify some attributes in the source model acting as dimension attributes, e.g. "publicationMentio", which actually provide data for multiple levels and dimensions in the requirements model shown in Figure 2. These attributes pack all the necessary information in a single database column, and must be transformed (Solvable Conflict) in order to obtain the necessary data for the DW. On the other hand, some dimensions like "Document" require combining multiple dimensions from the sources into a single one, as can be seen in Figure 8. In this case, we must combine the "Document_TD" and "Format" dimensions to obtain the two first levels in the "Document" dimension. As a result of the mapping, we identify the existence of an additional level, "Right", not considered in the requirements model which could provide useful information for the analysis.

[1]In order to focus on the main problem tackled in this work, we omit the relationships between attributes which have the same name in both schemata

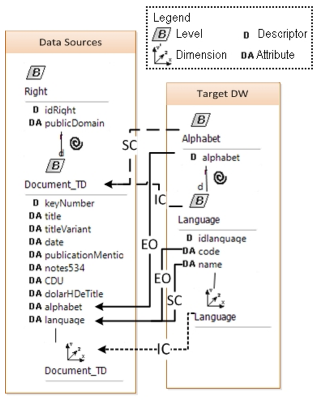


Figure 7: Irresolvable conflict between Document_TD and Language. We can only obtain alphabets from the data sources.

Furthermore, some attributes in the sources are empty, and the information is located in another member of the dimension. Finally, some attributes cannot obtain their value directly from the sources, but could be obtained by means of an external source, such as the country codes ("idCountry').

In addition to solve the name matching and structural problem, our approach allowed us to preserve traceability and perform requirements validation as well as to analyze the impact of changes. For example, the analysis of the relationships captured shows that, out of 32 attributes in the requirements model, only 23 attributes required a cleaning process, 8 attributes required a transformation to calculate their value, and 1 attribute was completely missing. More-

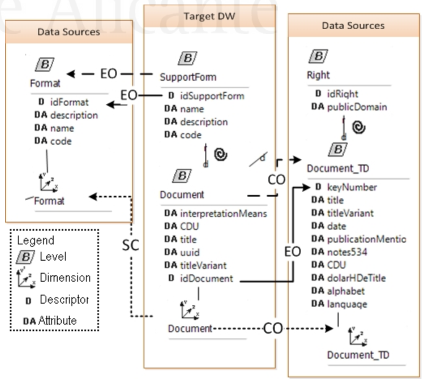


Figure 8: Relationships between pieces of work and their format

over, a critical attribute was identified in the data sources, providing information to 3 different descriptors in the requirements. Finally, one requirements level was not viable and its descriptor attribute had to be changed.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a formal approach that relates user concepts obtained from requirements to those coming from data sources, in order to better support the reconciliation process. Our approach is complementary to ETL processes, since it allows to easily identify and keep track of each element affected by a change both at data sources level as well as at requirements level. Thus the designer can easily identify how changes affect the different elements involved in the DW. Moreover, our approach also provides the necessary scaffolding to calculate a series of measures which may help in the analysis of alternative implementations, including but not limited to: (i) number of different sources that must be integrated to satisfy a given requirement, (ii) number of elements for which no information can be retrieved, and (iii) number of requirements supported by a given data source. Finally, we have shown the applicability of our approach by means of a real case study involving the integration process in the digital library of the University of Alicante. Thanks to our approach we were able to perform the analysis of the three aspects.

Our plans for the immediate future involve providing improved tool support for the traces. In the medium-long term, we plan to elaborate a series of metrics which are automatically calculated in order to evaluate the quality of the DW and the impact of changes.

6. ACKNOWLEDGMENTS

This work has been supported by the MESOLAP (TIN2010-14860) and SERENIDAD (PEII-11-0327-7035) projects from the Spanish Ministry of Education and the Junta de Comunidades de Castilla La Mancha. Alejandro Maté is funded by the Generalitat Valenciana (ACIF/2010/298).

7. REFERENCES

[1] Bonifati, A., Cattaneo, F., Ceri, S., Fuggetta, A., Paraboschi, S., et al. Designing data marts for data warehouses. *ACM Transactions on Software Engineering and Methodology 10*, 4 (2001), 452–483.

[2] Buneman, P., Khanna, S., and Wang-Chiew, T. Why and where: A characterization of data provenance. *Database Theory - ICDT 2001* (2001), 316–330.

[3] Chaudhuri, S., and Dayal, U. An overview of data warehousing and OLAP technology. *ACM Sigmod record 26*, 1 (1997), 65–74.

[4] Dayal, U., Castellanos, M., Simitsis, A., and Wilkinson, K. Data integration flows for business intelligence. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (2009), ACM, pp. 1–11.

[5] Eckerson, W. *Performance dashboards: measuring, monitoring, and managing your business.* Wiley, 2010.

[6] El Akkaoui, Z., and Zimányi, E. Defining ETL worfklows using BPMN and BPEL. In *Proceeding of the ACM twelfth international workshop on Data warehousing and OLAP* (2009), ACM, pp. 41–48.

[7] Giorgini, P., Rizzi, S., and Garzetti, M. GRAnD: A goal-oriented approach to requirement analysis in data warehouses. *Decision Support Systems 45*, 1 (2008), 4–21.

[8] Inmon, W. *Building the data warehouse.* Wiley-India, 2005.

[9] Kimball, R., and Ross, M. *The data warehouse toolkit: the complete guide to dimensional modeling.* Wiley, 2011.

[10] Kleppe, A., Warmer, J., and Bast, W. *MDA explained: the model driven architecture: practice and promise.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

[11] Luján-Mora, S., Trujillo, J., and Song, I.-Y. A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering 59*, 3 (2006), 725–769.

[12] Maté, A., and Trujillo, J. Incorporating traceability in conceptual models for data warehouses by using MDA. *Conceptual Modeling–ER 2011* (2011), 459–466.

[13] Maté, A., and Trujillo, J. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. *Information Systems 37*, 8 (2012), 753 – 766.

[14] Mazón, J.-N., Pardillo, J., Soler, E., Glorio, O., and Trujillo, J. Applying the i* Framework to the Development of Data Warehouses. In *iStar* (2008), pp. 79–82.

[15] Mazón, J.-N., and Trujillo, J. A hybrid model driven development framework for the multidimensional modeling of data warehouses. *SIGMOD Record 38*, 2 (2009), 12–17.

[16] Mazón, J.-N., Trujillo, J., and Lechtenbörger, J. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. *Data & Knowledge Engineering (DKE) 63*, 3 (2007), 725–751.

[17] Muñoz, L., Mazón, J.-N., Pardillo, J., and Trujillo, J. Modelling ETL Processes of Data Warehouses with UML Activity Diagrams. In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops* (2008), Springer, pp. 44–53.

[18] Romero, O., and Abelló, A. A framework for multidimensional design of data warehouses from ontologies. *Data & Knowledge Engineering 69*, 11 (2010), 1138–1157.

[19] Vassiliadis, P. *Data Warehouse Modeling and Quality Issues.* PhD thesis, Athens, 2000.

[20] Winkler, S., and von Pilgrim, J. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling 9* (2010), 529–565.

[21] Yu, E. *Modelling strategic relationships for process reengineering.* PhD thesis, Toronto, Ont., Canada, Canada, 1995.

5

Improving User Requirement Diagrams

The content of this chapter corresponds with the following papers:
Maté, A. *and Trujillo, J. and Franch, X. Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star. Journal of Systems and Software (**JSS**). (In Press) Impact Factor: 1.135*

Data warehouse development requires communicating with users in order to elicitate their needs and identify the relevant information to be stored. Current approaches have evolved from text-based requirements into goal models in an effort to improve the communication between data warehouse designers and decision makers. However, current goal diagrams do not provide any mechanisms to be partitioned. In practice, the size of data warehouse requirements models can increase dramatically, going over hundreds of elements. In turn, it becomes more difficult to use these diagrams as a mechanism of communication and to focus in specific parts of the diagram. Therefore, in the third part of this PhD Thesis we focus on (i) defining mechanisms to partition the diagrams with specific semantics, and (ii) evaluate how this partition affects the design and usability of the diagrams.

Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star

Alejandro Maté[a,*], Juan Trujillo[a], Xavier Franch[b]

[a]*Lucentia Research Group, Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n - 03690 San Vicente del Raspeig - Alicante, Spain*
[b]*BarcelonaTech, Universitat Politècnica de Catalunya, Calle Jordi Girona, 31 - 08034 Barcelona, Spain*

Abstract

The success rate of data warehouse (DW) development is improved by performing a requirements elicitation stage in which the users' needs are modeled. Currently, among the different proposals for modeling requirements, there is a special focus on goal-oriented models, and in particular on the i* framework. In order to adapt this framework for DW development, we previously developed a UML profile for DWs. However, as the general i* framework, the proposal lacks modularity. This has a specially negative impact for DW development, since DW requirement models tend to include a huge number of elements with crossed relationships between them. In turn, the readability of the models is decreased, harming their utility and increasing the error rate and development time. In this paper, we propose an extension of our i* profile for DWs considering the modularization of goals. We provide a set of guidelines in order to correctly apply our proposal. Furthermore, we have performed an experiment in order to assess the validity our proposal. The benefits of our proposal are an increase in the modularity and scalability of the models which, in turn, increases the error correction capability, and makes complex models easier to understand by DW developers and non expert users.

Keywords: Data Warehouses, modules, user requirements, i-star

1. Introduction

Organizations manage huge amounts of information, and wish to take informed decisions by using that information. Nowadays, there is an increasing importance of the Business Intelligence (BI) in the enterprise environment. In

*Corresponding author. Tel: +34 96 5909581 ext. 2737; fax: +34 96 5909326
Email addresses: amate@dlsi.ua.es (Alejandro Maté), jtrujillo@dlsi.ua.es (Juan Trujillo), franch@essi.upc.edu (Xavier Franch)

fact, the Gartner Group showed that, during the recent recession period, the BI market not only did not decrease, but instead it grew a 4% [1].

At the core of the BI, among other technologies, is the Data Warehouse (DW). DWs integrate several heterogeneous data sources in multidimensional structures (i.e. facts and dimensions) in support of the decision-making process [2, 3]. Therefore, the development of the DW is a complex process which must be carefully planned in order to meet user needs. This process can be even more complex, if we consider that requirements for the DW change as the organization's information needs change. For this reason, the modeling of user needs is a very important aspect of DWs, which can be accomplished by means of goal-oriented models. These models represent the users' intentions in a requirements model using goals and are easily understandable by users. Among the goal-oriented approaches, the i* framework [4], is currently one of the most widespread goal modeling frameworks. This framework has been applied for modeling organizations and system requirements among others.

However, due to the idiosyncrasy of DWs, a specialization of the i* framework was required, in order to correctly model the desired information goals. In our previous work, we presented the required specialization, along with our development methodology for DWs. In our proposal, we follow the Model Driven Architecture (MDA) [5], starting from a Computation Independent Model (CIM) layer where requirements are modeled. From this layer, the DW schema is derived into a Platform Independent Model (PIM) layer, reconciliated with the information present in the data sources, and derived into its implementation. Therefore, the CIM layer is crucial, since it acts as the starting point of the process.

Nevertheless, as pointed in [6], the i* framework lacks scalability due to the absence of modularity. Modularity is a well-known concept in software engineering. As far as the start of the 70s, modular programming became a hot topic and the benefits of splitting complexity using some well-defined criteria were subject of several seminal papers [7]. Afterwards, modularity spread over other life-cycle activities and artifacts, and became very popular especially in the context of system design, where the notion of decomposing a system into its parts offers several benefits like better flexibility, management and testability, to name a few. Since we are interested in modularity applied to specification models, it can be defined as the ability to decompose a large model into several sub-models, such that they independently have a well-defined meaning, and whose combination solves the original problem.

Since the work presented in [8] is a specialization of the original framework, it lacks modularity as well. As DW requirements models may become very complex, this hurts their readability and comprehension, becoming more difficult to correct and update as requirements change. We have experienced this drawback ourselves, as some of our real projects had over 16 goals, 15 tasks and 53 resources for a single actor. These models became huge for correction and communication with the users. Sometimes these models even included repeated DW elements in the same model with different structure, since designers forgot which elements were already defined. Therefore, it is important to improve this aspect in order to manage corrections and changes in DW requirements in an easier way.

In the short version of this paper [9] we proposed an extension of our i* profile [8], in order to adapt it and improve its modularity. In turn, this increases its manageability, as well as the comprehension capability of the user when dealing with complex models. With these modifications, the communication between users and developers is improved, leading to higher success rates. We also provided a set of guidelines to correctly apply the proposal. Moreover, we performed an experiment in order to assess the validity our proposal.

In addition, in this long improved version, we (i) include the definitions of the main goals (strategy, decision, and tactic) in which we classify the final user's needs, (ii) perform an ontology mapping between concepts in the i* framework and the DW context, and include *Decision*, *Information*, and *Hierarchy* modules, increasing the scalability of the models, (iii) include an extended case study in order to show the applicability and benefits of applying our proposal, as well as (iv) describe a second, deeper analysis of the results, reaching new conclusions to better define and organize the modules.

The rest of the paper is structured as follows. Section 2 presents the related work in this area. Section 3 presents our i* profile for DWs. Section 4 proposes the different types of modules for our i* profile. Section 5 presents an example of application and the experiment performed. Finally, Section 6 summarizes the conclusions and future work.

Basic knowledge of i* is assumed in the paper, see [4] and the i* wiki (http://istar.rwth-aachen.de) for a thorough presentation.

2. Related Work and Background

Scalability is probably the best-known and widely acknowledged problem of i*. It is fact that i* models quickly grow in size (see [10] for an illustrative example of large-scale model) making them rapidly difficult to manage. As argued by [11], the scalability problem is a direct consequence of the lack of mechanisms for modularization. In that work, the authors conducted an empirical study on different aspects related to i* as a modeling language, and it was concluded that modularity is not supported in i*, consequently we may say that scalability is not supported either. Since the core of the language has not evolved since then, the problem persists nowadays.

When dealing with scalability issues, other works have focused on i* modularity in general, like the one in [6]. However, these modules do not have meaningful semantics for being applied in DWs, which could favor the understandability of the modularization process. Therefore, before performing any kind of adaptation, a study of the target domain must be performed along with a mapping between the concepts. Then, the necessary modules should be defined accordingly to how the target domain is structured. In other areas, a similar approach has been applied successfully in order to improve the scalability of the models. For example, in [12] the authors introduce new elements in the notation of task models that summarize several elements in the diagrams, allowing the designers to manage their complexity while keeping the models meaningful.

Within the area of DW requirements modeling, initial works such as [13] propose to represent DW requirements by means of use case diagrams. Use cases divide DW requirements into an actor dependency diagram and several use case specifications. However, use case notation is difficult for users to understand. Therefore, more recent works focus on representing DW requirements in terms of goals, both i* based, such as [8, 14], and non-i* based [15]. The i* based approaches suffer from the lack of modularity intrinsic to the i* core, as they do not provide any mechanisms to control the complexity and size of the diagrams. Unfortunately, non-i* based models do not include any modularization elements either, thus the complexity and size of the diagrams is only determined by the complexity of user requirements themselves.

In our previous work, we developed a UML profile for modeling DWs at conceptual level [16], where the importance of packages was shown, in order to improve the modularity of DW conceptual models. The packages included were StarPackage, for differentiating cubes, DimensionPackage, for aggregating dimensions along with their hierarchies, and FactPackage, which included the associated fact. These packages allow the developer to analyze the model at different levels of detail, hiding those elements on which he has no interest, lowering the complexity of the model and increasing its readability. In turn, this aspect makes the developing of the schemata less error prone.

However, since the conceptual level is closer to developers than to users, we required models with a higher level of abstraction in the development process. Therefore, in order to improve the communication with the users, and increase the success rate of DW projects, we included a RE phase in our methodology [8]. In this RE phase, requirements are captured on a model by using a UML profile [8] based on the i* framework [4]. From these requirements, the conceptual model is automatically derived by means of Model Driven transformations [17], transforming the different Business Process, Contexts, and Measures associated with the goals at requirements level into Facts, Dimensions and Measures at the conceptual level.

Nevertheless, although our i* profile incorporated the necessary semantics and methodology, it lacks any kind of modularity. In turn, this hurts the communication with the users, since complex requirements models can become huge and difficult to read and understand. Now, in this work, we complement our approach, by improving the modularity and scalability of our i* profile. We include modules for the decision and information goals, as well as for hierarchies of contexts. By improving the modularity, the models are easier to read, which, in turn, reduces the error rate and increases user satisfaction.

3. i* Profile for DWs

Our i* profile, presented in [8], follows a Goal-Oriented Requirements Engineering (GORE) approach. GORE is concerned about modeling goals, thus obtaining user requirements by following a refinement process [18]. The i* modeling framework [4] provides mechanisms to represent actors, their dependencies, and structuring the business goals that organization pretends to achieve. This framework establishes two models: the strategic dependency (SD) model for describing the dependency relationships among various actors in an organizational context, and the *strategic rationale* (SR) model, used to describe actor interests and concerns, and how they might be addressed. From now on, we focus on the SR models to model goals and information requirements of decision makers.

The first step is aligning the ontology of i* with the target domain. In the DW domain, the requirements model specifies the informational needs of different stakeholders in order to support the decision-making process. This information is used to improve the performance of a business activity. In our observation, several types of goals which arise naturally during the design process.

- **Strategic goals**. They represent a desired change from a current situation into a future one. A strategic goal is always related to the main objectives of the business process (see below) that is being improved. Therefore, strategic goals always have an objective to be met, either clear, i.e. Sales increased, or fuzzy, i.e. Number of clients significantly increased. They are long-term goals that cause an immediate benefit for the organization when fulfilled.

- **Decision goals**. They operationalize strategic goals into appropriate actions by answering the question: "how can a strategic goal be achieved?". Decision goals represent decisions that make use of information in order to provide a benefit for the organization. Decisions can be described either in terms of objectives, i.e. Some kind of promotion offered, or in terms of tasks, i.e. Open new stores. The benefit obtained by a decision goal is directly related to the achievement strategic goals by means of the decision goal.

- **Information goals**. They identify the information required for a decision goal to be achieved by answering the question: "how can decision goals be achieved in terms of information required?". Information goals specify the necessary information to be gathered, typically by means of an analysis. Therefore, they can be defined in terms of goals, i.e. Customer purchases analysed, or in terms of the analysis process, i.e. Examine the stocks daily. The satisfaction of information goals allows decision makers to take decisions and fulfill decision goals.

These three types of goals have a decreasing level of abstraction, from strategic (most) to informational (less). In addition, there is a contextualization relationship among goals: decision goals only take place within the context of strategic goals, and informational goals only take place inside the context of decisional goals.

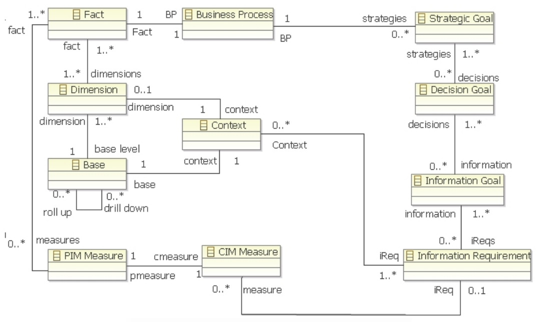


Figure 1: DW domain metamodel

Along with these goals, the DW domain includes several key concepts related to the multidimensional level of DWs [19]. The description of these concepts, presented in Figure 1, is as follows:

- **Business Processes**. Represent an activity that the user wishes to improve by means of strategies. These business processes have associated a series of performance indicators, represented as measures of the business process. Sometimes business processes can be described in terms of the goal pursured by the activity, i.e. Contracts agreed, or in terms of the activity itself, i.e. Make sales.

- **Information Requirements**. Represent the necessary information in order to achieve an information goal. They are always considered in terms of information gathering tasks. Information requirements are decomposed into context and measures, that represent the information to be gathered by the information requirement.

- **Contexts**. Describe the necessary additional data in order to analyze a given business process. They represent information about entities involved in the business processes of the organization, i.e. Department or Customer, and can be grouped into hiearchies, i.e. Customers within the same city.

- **CIM measures**. Represent indicators of performance of a business process. They provide quantitative information that can be assessed by decision makers in order to evaluate if business processes are performing as expected. PIM measures are the multidimensional counterpart of indicators.

- **Bases**. Represent the multidimensional counterpart of contexts. They describe the levels of aggregation within a dimension of the data warehouse.

Table 1: Alignment of the DW concepts with the i* framework

DW	i* concept	Example
Strategic goal	Goal	Sales increased
	Softgoal	Number of clients significantly increased
Decision goal	Goal	Some kind of promotion offered
	Task	Open new stores
Information goal	Goal	Customer purchases analysed
	Task	Examine stocks daily
Business Process	Goal	Contracts agreed
	Task	Make sales
Information Requirement	Task	Record task assignments and durations
Context	Resource	Market; Department
Measure	Resource	Discount; Income generated

- **Dimensions**. Represent a context of analysis to analyze a fact, and are formed by sets of hierarchies. Each hierarchy can have one or more groups of bases, forming classification and generalization hierarchies in the multidimensional model and defining the structure of the data warehouse.

- **Facts**. Represent the multidimensional counterpart of the business process which wants to be improved.

After having presented the target domain concepts, we proceed to map the DW concepts with the i* ontology, as shown in Table 1. As can be perceived, not all the elements are aligned. Specifically, the concepts of dimension, fact, base, and PIM Measures, are not considered part of the requirements engineering process, thus they are left aside as external elements. Moreover, some DW concepts can be mapped into more than one i* intentional type depending on the level of abstraction and the cut criterion chosen.

Once we have defined and mapped the concepts in our i* profile, we will describe them through an example, shown in Figure 2. In this example, we start the requirements analysis from a business process (BP), related to the decision-maker. The BP, which is the center of the analysis, models an activity of interest for the decision-maker. In this case the activity is to *Make Contracts*, and has associated a series of strategic goals, aimed to improve the business performance. Strategic goals represent the highest level of abstraction. They are thought as changes from a current situation into a better one in terms of business process objectives. In our case, the strategic goals associated with the BP are *Cost of contracts minimized* and *Quality of workers increased*. Other examples of strategic goals would be *Increase sales*, *Increase number of customers*, *Decrease cost*, etc. Their fulfillment causes an immediate benefit for the organization.

In order to achieve these strategic goals, there are a series of decision goals that must be met. Decision goals represent the medium level of abstraction in our SR models. They try to answer the question "how can a strategic goal be

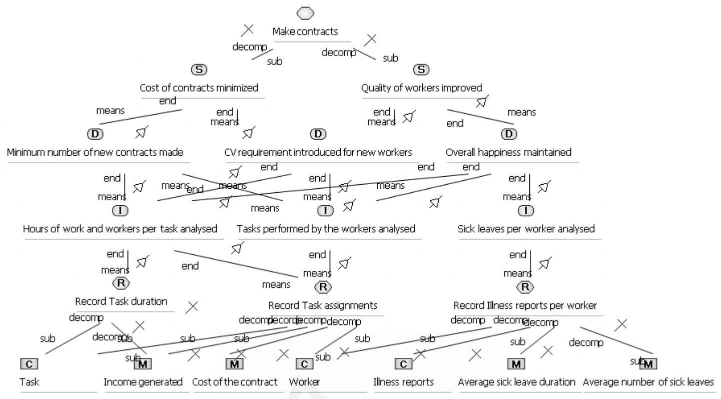

Make contracts
decomp
sub
decomp
sub
Cost of contracts minimized
Quality of workers improved
end
means
end
means
end
means
end
means
Minimum number of new contracts made
CV requirement introduced for new workers
Overall happiness maintained
end
means
end
means
end
means
end
means
end
means
end
means
Hours of work and workers per task analysed
Tasks performed by the workers analysed
Sick leaves per worker analysed
end
means
end
means
end
means
end
means
Record Task duration
Record Task assignments
Record Illness reports per worker
decomp
sub
decomp
sub
Task
Income generated
Cost of the contract
Worker
Illness reports
Average sick leave duration
Average number of sick leaves

Figure 2: Example of the current monolithic CIM representation

achieved?", and they aim to take the appropriate actions to fulfill a strategic goal. They are related to strategic goals by intentional means-end relationships. In our example, in order to achieve *Cost of contracts minimized*, it has been decided that it is necessary to have the *Minimum number of new contracts made* as well as have a *CV requirement introduced for new workers*, in order to achieve the strategic goal. However, decision goals can affect more than one strategic goal. In our case, the last decision goal is related with *Quality of workers increased* strategic goal as well, since the CV affects the quality of the new workers being employed. Other examples of decision goals would be *Determine some kind of promotion* or *Open new stores*. Their fulfillment only causes a benefit for the organization if it helps to reach strategic goals, since decision goals only take place within the context of strategic goals.

As with the strategic goals, the decision goals can be achieved by having the necessary information available. This required information is modeled by means of the informational goals. Information goals represent the lowest level of abstraction. They try to answer the question: "how can decision goals be achieved in terms of information required?", and they are related to the information required by a decision goal to be achieved. In our example, the information required is *Hours of work and workers per task analysed* and *Tasks performed by the workers analysed* for each decision goal, whereas the information about *Sick leaves per worker analysed* affects only the *Overall happiness maintained* decision goal. Other examples of information goals are *Analyze customer purchases* or *Examine stocks*. Their fulfillment helps to achieve decision goals and they only happen within the context of decision goals.

Finally, informational goals are achieved by means of information requirements. In our case, we need to *Record Task duration*, *Record Task assignments*, and *Record Illness reports per worker* in order to gather the required informa-

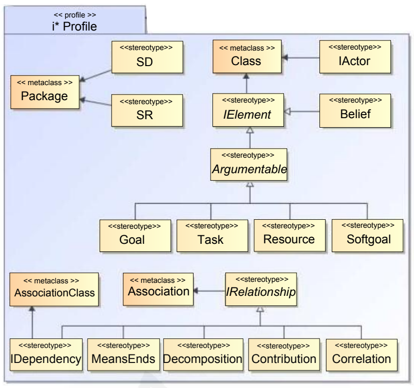


Figure 3: Original i* elements

tion. Each of these requirements is decomposed into contexts and measures. The example includes the *Task, Worker*, and *Illness report* contexts as well as the *Income generated, Average sick leave duration* and *Average number of sick leaves* measures, which determine the performance of the business process.

As has been shown in the example, a lower-level goal can be a part of different higher-level goals. This process is repeated at all levels, leading to highly interrelated elements in the model, making difficult to comprehend the business strategy in huge models. In order to solve this issue, we propose a series of modules, packaging all the elements related to a given higher-level goal on each module.

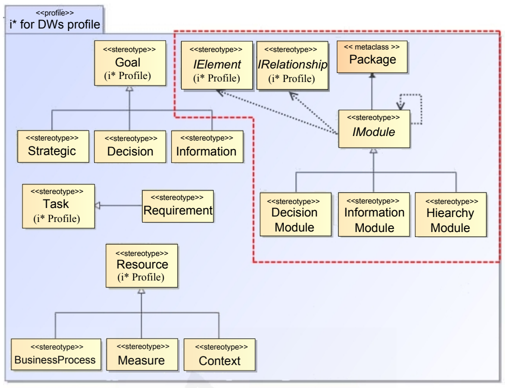
4. Definition of Modules and Guidelines

In this section, we will present the extension to our i* profile for DWs, by defining the proposed modules and the extended metaclasses. Furthermore, we will also present some guidelines to the application of the modules proposed.

4.1. Definition of Modules

First, we will define our proposed modules, in order to manage the complexity of the goal models. The modules which we will define are strongly related to the concepts identified in the DW domain. Therefore, each module has a specific semantic associated adapted for the DWs. We have not included a module for strategic goals since typically there is only a few of them.

- **Decision modules** include the elements related to a given *decision goal*. They can include *decision goals, information goals, requirements, contexts, measures, other decision modules, information modules,* and *hierar-*



<<profile>>
i* for DWs profile
<<stereotype>> Goal (i* Profile)
<<stereotype>> IElement (i* Profile)
<<stereotype>> IRelationship (i* Profile)
<< metaclass >> Package
<<stereotype>> IModule
<<stereotype>> Strategic
<<stereotype>> Decision
<<stereotype>> Information
<<stereotype>> Task (i* Profile)
<<stereotype>> Requirement
<<stereotype>> Decision Module
<<stereotype>> Information Module
<<stereotype>> Hiearchy Module
<<stereotype>> Resource (i* Profile)
<<stereotype>> BusinessProcess
<<stereotype>> Measure
<<stereotype>> Context

Figure 4: i* profile with modules extension for DW

chy modules. They contain all the necessary information to take a given decision, which helps achieving a *strategic goal*.

- **Information modules** include the elements related to a given information goal. They can include *information goals, requirements, contexts, measures, other information modules,* and *hierarchy modules*. They aggregate all the information which is necessary to satisfy a given information goal.

- **Hierarchy modules** include the elements which constitute a hierarchy. They are formed by the different contexts which represent the different levels of aggregation of a dimension. They can only include *contexts*. These modules help with the reusability of the dimensions at the requirements level, and hide the complexity of hierarchies when it is unnecessary.

These modules are shown in Figure 4 together with the rest of the elements in the i* for DWs profile. The modules defined are loosely coupled with the core i* elements, shown in Figure 3, and extend from the *Package* element. Moreover, they include an intermediate element, *iModule*, in order to help with the definition of OCL constraints that guarantee their correct application in CASE tools. After having defined the modules, we will present a set of guidelines to apply them while minimizing the drawbacks.

4.2. Guidelines

In this section, we will give some guidelines to use the provided modules, in order to maximize their benefits. It is not mandatory to package every element, although it is recommended for the sake of uniformity and to provide different abstraction levels, which results in a more intuitive approach (G1). However, if some parts of the goal tree have a low complexity, it might not be necessary to group them in a separate package (G2). For each package created, there should be a single root element, corresponding to the type of package, which acts as a connection for higher level elements. This element should have no dependencies to other elements inside the same package (G3). The name of the package should be the same as the root element, in order to help with the identification of the corresponding packaged subtree (G4). For each decision goal a *Decision module* should be created (G5). Inside a decision module there should be an *Information module* for each information goal that supports the decision goal (G6). If included in a CASE tool, elements should not be repeated, but instead imported from packages where they were first defined (G7). *Information modules* should include all the elements related to the information goal, importing elements where necessary, and always including a *Hierarchy module* for each different hierarchy of contexts present (G8). These *Hierarchy modules* represent the lowest level of abstraction in the strategic rationale, and should be always separated from the goal tree, in order to hide the details of the hierarchies of contexts unless they are necessary (G9).

4.3. Improvements in Scalability

In order to demonstrate the improvements in scalability obtained with the introduction of modules it is first necessary to define the concept of scalability. Scalability is a term often used intuitively, but with no clear definition. When used in the context of software engineering notations and diagrams [20], it usually refers to "the property of reducing or increasing the scope of methods, processes, and management according to the problem size [...] Inherent in this idea is that software engineering techniques should provide good mechanisms for partitioning, composition, and visibility control. It includes the ability to scale the notation to particular problem needs, contractual requirements, or even to budgetary and business goals and objectives." In practice, scalability problems arise typically when models become too large to be handled adequately, as shown in the previous sections.

In the particular case of i*, inherited by i* for DWs, the lack of modularization elements limits the capability of the designer to partition the model, control the visibility of elements, and scale the notation to particular needs, i.e. decision maker's goals vs. data warehouse structural requirements. The modules proposed in this section enable the designer to (i) perform partitioning, dividing a single diagram into multiple ones, thus reducing the visual complexity of each individual sub-diagram, (ii) control the visibility of unnecessary elements, by hiding lower abstraction goals and their structure by means of packages, and (iii) manage the scope of the diagrams, by separating decision maker's goals from data warehouse structural requirements (contexts and measures) derived from these goals.

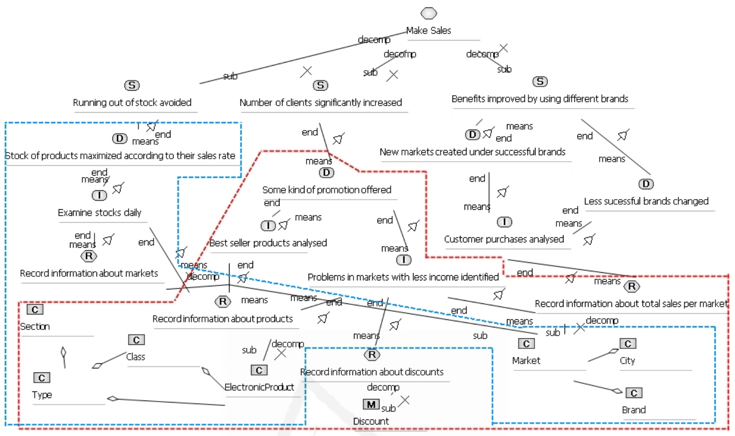

Make Sales
decomp
sub
Running out of stock avoided
Number of clients significantly increased
Benefits improved by using different brands
Stock of products maximized according to their sales rate
Examine stocks daily
Record information about markets
Some kind of promotion offered
Best seller products analysed
New markets created under successful brands
Less sucessful brands changed
Customer purchases analysed
Problems in markets with less income identified
Record information about total sales per market
Record information about products
Record information about discounts
end
means
Section
Class
Type
ElectronicProduct
Discount
Market
City
Brand

Figure 5: Part of the requirements model for Sales analysis with the scope of the decision goal *Some promotion offered* marked in red, and the scope of *Stock of products* in blue

5. Example of Application and Experiment Results

In this section we will present the application of our proposal to an example, as well as the results of two experiments performed in order to analyze how users and developers perceive the modularized models.

5.1. Example of Application

The following example presents a simpler goal tree, as opposed to the one presented in Section 3, whereas the contexts and hierarchies are better defined at requirements level than previously, and the scope of each element may be hard to identify. In this case, the contexts can be aggregated at different levels of detail, presenting market and electronic product contexts as the lowest level, which can be aggregated up to state and section levels.

This example can be modularized using the proposed packages, decreasing its complexity and providing different levels of detail. In this sense, the application of modules results in a first level providing an overview of the strategies related to the business process and their corresponding decisions. In this case, the goal tree presents the different decision packages as its leaves, which are further detailed in their corresponding models. Figure 6 presents the previous business process with 3 related strategies and the corresponding 3 decision packages.

For each decision, we have a different package which includes their related information goals and presents the intermediate level of detail. The elements corresponding to each information goal are also modularized in their own packages, which are the leaves of the decision models.

Finally, for each information goal, we have a package which includes their related information requirements, presenting the corresponding hierarchy packages and measures. Figure 7 presents a information goal with 3 different information requirements, which account for a total of 2 measures and 2 hierarchies.

In this model, hierarchies are included at CIM level, but are separated into their own packages. This hides the lowest level of detail, which acts as a bridge between the requirements and conceptual models, whenever it is necessary, allowing us to focus on the modeling of the goals and their related elements.

5.2. Experiment Results

We have performed two experiments, with participants ranging from non-expert people to DW designers and experts on i* modeling, in order to evaluate the impact of our proposal. These experiments are part of a family of experiments for assessing the validity and impact of the proposal, following the same methodology as in [21]. There were two rounds of experiments. The first round presented two examples, one smaller (see Figure 2, *Example 1*) and one bigger (see figure 5, *Example 2*). Both examples were presented in generic i* notation with our own stereotypes, in order to make the questionnaires more accessible for all the participants. Monolithic models were presented in a single sheet, whereas modularized models were presented in multiple sheets. Both base examples were small in comparison with real project models, presenting fewer goals and contexts in order to make them manageable. These examples were presented in the four combinations:

- Questionnaire 1.A: Example 1 without modules, Example 2 with modules.
- Questionnaire 1.B: Example 2 with modules, Example 1 without modules.
- Questionnaire 2.A: Example 2 without modules, Example 1 with modules.
- Questionnaire 2.B: Example 1 with modules, Example 2 without modules.

A total of 28 participants filled the questionnaire. Each participant was given one kind of questionnaire and they were asked to identify and mark a series of elements on each model (which were the same for both modularized

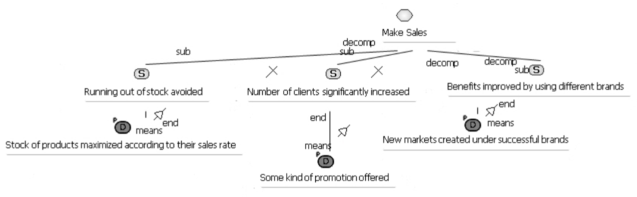

Make Sales
decomp
sub
sub
decomp
decomp
sub
Running out of stock avoided
Number of clients significantly increased
Benefits improved by using different brands
end
means
Stock of products maximized according to their sales rate
end
means
Some kind of promotion offered
end
means
New markets created under successful brands

Figure 6: Strategy level for Sales analysis

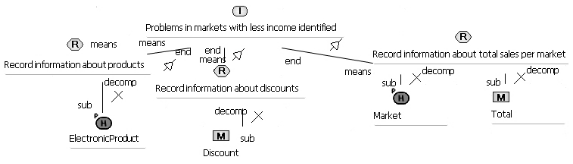

Problems in markets with less income identified
means
end
end
means
end
means
Record information about products
Record information about discounts
Record information about total sales per market
decomp
sub
ElectronicProduct
decomp
sub
Discount
decomp
sub
Market
decomp
sub
Total

Figure 7: Information level for Sales analysis

and monolithic versions of the same example). The participants were not able to modify their answer once they finished a question. After completing the identification tasks on each model they were asked to give scores for a series of characteristics of the model, ranging from 0 to 3. Finally, after having finished identifying elements in both models, they were asked abstract questions about how they would add a new element at different levels (decision goal, information goal and a set of contexts), while not referencing any example. The group of participants was formed by 14 self-evaluated beginners, 8 participants with some experience and 6 participants were advanced users/experts. Regarding the accumulated experience, 17 participants had less than 1 year of experience in the i* framework, 8 participants had between 1 and 5 years of experience and 3 participants had over 5 years of experience. Given this sample, the hypothesis for our experiment are:

Null hypothesis, H_{0_1}: There is no statistically significant correlation between the modularization of models and the time required for different tasks and the characteristics perceived.

Hypothesis H_{1_1}: $\neg H_{0_1}$

The independent variables in the experiment are those whose effects should be evaluated. In our experiment, this variable corresponds to how the model is structured (modularized or monolithic). On the other hand, dependent variables for the experiment are the understandability and manageability of the models, evaluated accordingly to the time necessary to perform different tasks on the models. As there was no statistically significant correlation between the structure of the models and most tasks, we did not calculate further values like efficiency and effectiveness. Therefore, the results will be discussed in terms of trends. The experiment results for the first round are shown in Table 2. Time is measured in seconds.

In order to obtain the results, in every step, first, outliers were identified and filtered. Then, the second step was to perform a variance analysis of the data (ANOVA), in order to identify significant differences between the models. After the first step, 27 questionnaires were left, which were used for the statistical analysis. The significance analysis ($\rho < 0.05$) revealed that the reading time for the Sales model was significantly different (inferior) than when built in a monolithic way. The only other significant difference perceived was the scalability of both examples, which had a notably increase.

Table 2: Tasks performed (left) and independent (top) variables for experiment 1

	Monolithic	Modularized	ρ
Avg. reading time Sales	299.31	210.31	0.037
Identif. task 1 Sales	190.08	278.62	0.074
Identif. task 2 Sales	190.94	165.08	0.396
Avg. reading time Contracts	162.73	181.33	0.576
Identif. task 1 Contracts	150.07	211.5	0.112
Identif. task 2 Contracts	124.33	161.00	0.096
Avg. errors per questionnaire Sales	0.82	0.47	0.247
Avg. errors per questionnaire Contracts	0.33	0.36	0.906
Readability score Sales	2	1,93	0.826
Scalability score Sales	1,41	2,26	0.016
Comprehension score Sales	1,5	1,87	0.229
Modifiability score Sales	1,5	2,06	0.079
Readability score Contracts	2,27	2,33	0.803
Scalability score Contracts	1,67	2,41	0.011
Comprehension score Contracts	2,13	2,05	0.857
Modifiability score Contracts	1,73	2,17	0.128

However, we perceive an increase in time spent in order to identify and mark elements. The identification tasks required to identify and mark all elements related to a decision goal (task 1) and only the lowest level (contexts and measures) elements related to another goal. This increase in time can be due to marking a higher number of elements in 4 different sheets, as opposed to a single sheet in the monolithic model. Nevertheless, the number of wrong answered questions notably diminished in the Sales example when it was modularized. This is specially relevant, since participants identifying elements in the modularized example had to correctly identify the detail level of a package in the next sheet, whereas those in the monolithic example did not suffer from this drawback. Even though, participants identifying elements in the monolithic Sales model systematically forgot different contexts and measures.

Finally, when asked about how they would structure the models, most of the participants chose to organize them in a modularized fashion. Out of 27 participants 17 chose to package the decision goals and their related elements, 16 packaged the information goals, and 19 chose to package a new hierarchy and include it inside another package wherever it was necessary. It is noteworthy that, although we also analyzed the results according to the participants' expertise and years of experience on the i* framework, no significant correlation nor trend was found that differentiates both groups. This suggests that the addition of modules may affect participants similarly without regards to their experience.

After the first round, we performed a second round with 21 participants, including modification tasks over existing models, as well as the creation of a new model. The examples were the same as in the previous round, and they were presented in the same fashion. The group of participants in this second

Table 3: Tasks performed (left) and independent (top) variables for experiment 2

	Monolithic	Modularized	ρ
Modif. task 1 Sales	202	154,27	0.327
Modif. task 2 Sales	223,6	290	0.217
Modif. task Contracts	128,73	197,6	0.002
Avg. Time drawing	1306,67	1891,44	0.019
Avg. Time/element	50,10	44,34	0.809
Avg. number of elements	25,67	42,89	0.000
Avg. unique non package elements	25,67	27,67	0.021

round was formed by 9 self-evaluated beginners, 5 participants that had some experience, and 5 participants who were advanced users/experts. Furthermore 2 participants did not provide details about their background. As in the previous case, no effect of the participants' expertise on the results was found. The results are shown in Table 3.

After the first step, 4 questionnaires were excluded for the modification tasks, leaving a total of 17 questionnaires. However, the statistical analysis did not show any significant differences in the modification tasks. As previously, we can also perceive a decrease in time spent when the model is bigger and we perform small modifications on a single module (task 1), whereas there is an increase when we require information from multiple modules (task 2).

Finally, the creation of a new model from the scratch had a sample of 15 questionnaires, with a significant correlation between the structure of the model created and every result. Time spent was notably superior for models created with a modularized approach while time spent per element drawn was inferior when the model was modularized. Most importantly, the average number of elements identified from the text which described the model was superior when modules were applied, as opposed to the monolithic structure. Additionally, some monolithic models (filtered in the outliers analysis), presented repeated elements, which should not be created, and tend to increase in number as the model gets bigger.

A second, more detailed review of the results revealed an interesting result. Table 2 shows the number of questions incorrectly answered (average number of errors), either because of a single mistake or because of multiple mistakes. When analyzing the exact nature of errors, we found out that participants using monolithic models mixed different branches of the goal tree and overlooked several elements. On the other hand, participants using the modularized models overlooked sistematically overlooked measures when they were requested to identify hierarchies and measures. This is a significant result since, currently, information modules include both goals and DW elements. When participants tried to locate DW elements, they kept going deeper in the diagram until they found only DW elements, i.e. hierarchy modules. Thus, they systematically forgot measures in the answer.

According to these results, we intend to redefine the modules, substituting information modules with information requirement modules, that are more focused on DW elements. This way, we expect participants to have an easier time identifying where DW elements and user goals are located in the diagram. Furthermore, this redefinition also reduces the amount of diagrams required to interact with the users in order to validate the goals within the goal tree.

6. Conclusions and Future Work

Traditionally, i* models lack any modularity, suffering from scalability and readability issues. Regardless its widespread adoption, the i* framework is still facing several open issues. For instance, [22] mentions as directions of further research: clear definition of the i* language core, proposal of modeling methodologies and analysis techniques, and proposal of modularity constructs. Lack of modularity has been reported to harm model scalability and readability.

Therefore, although the profile presented in [8] is adapted for the semantics present in DWs, it suffers from the same issues as the original framework, since it provides no additional modularity. In turn, when real project models become huge, they turn from a useful tool for communicating with the user into a burden which requires too much work to correct, use and modify. Therefore, an improve in modularity is required in order to maintain the quality of the requirement analysis for DWs, while maintaining the specific semantics for them.

In this work, we have presented a proposal for applying modules, specially designed for DWs. We have defined our proposal, and provided some guidelines on how to correctly apply it. We also have shown an example of application and we have performed an experiment, with our proposal, including participants ranging from new users to experts on i* modeling. The results show a significant increase the scalability of the models, as well as a reduced error rate when identifying the scope of an element present in the model, while helping to create richer goal models. We also perceived an increase in the time necessary to perform different tasks over the models, which may be reduced if these tasks were performed with a CASE tool. Finally, experiment results show that most people tend to group elements in packages at different levels of abstraction, as opposed to adding them in a global schema, which may be helpful in the communication of models.

According to the new findings obtained from a deeper analysis, it would be positive to redefine the modules by substituting the intermediate level (information) by a lower abstraction level that completely separates user goals from information itself (information requirement). This way, both DW designers and users can focus strictly on analyzing the validity of the current set of goals or the current information considered to make decisions.

Finally, an interesting research direction is to analyze how the set of modules could be further extended to consider advanced types of data warehouses, such as distributed spatio-temporal DWs and stream DWs [23, 24, 25, 26]. In these cases, the special nature of the information stored should be considered, allowing not only to better package their requirements, but also improving the maintainability and change management of the data warehouse structure.

Acknowledgments.

This work has been partially supported by the ProS-Req (TIN2010-19130-C02-01) and by the MESOLAP (TIN2010-14860) and SERENIDAD (PEII-11-0327-7035) projects from the Spanish Ministry of Education and the Junta de Comunidades de Castilla La Mancha respectively. Alejandro Maté is funded by the Generalitat Valenciana under an ACIF grant (ACIF/2010/298).

[1] G. Group, Gartner Group BI Revenue Analysis 2009, `http://www.gartner.com/it/page.jsp?id=1357514`.

[2] W. Inmon, Building the data warehouse, Wiley-India, 2009.

[3] R. Kimball, The data warehouse toolkit, Wiley-India, 2009.

[4] E. S.-K. Yu, Modelling strategic relationships for process reengineering, Ph.D. thesis, Toronto, Ont., Canada, Canada (1995).

[5] A. Kleppe, J. Warmer, W. Bast, MDA explained: the model driven architecture: practice and promise, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

[6] X. Franch, Incorporating Modules into the i* Framework, in: CAiSE, Vol. 6051 of LNCS, Springer Berlin, 2010, pp. 439–454.

[7] D. L. Parnas, On the criteria to be used in decomposing systems into modules, Communications of the ACM 15 (12) (1972) 1053–1058.

[8] J.-N. Mazón, J. Pardillo, J. Trujillo, A model-driven goal-oriented requirement engineering approach for data warehouses, ER'07, Springer-Verlag, 2007, pp. 255–264.

[9] A. Maté, J. Trujillo, X. Franch, A modularization proposal for goal-oriented analysis of data warehouses using i-star, in: Proceedings of the 30th international conference on Conceptual modeling, ER'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 421–428.

[10] J. Lockerbie, N. A. Maiden, A. Dotan, V. Lichtner, Using i* to support a summative evaluation, in: 4th International Workshop on iStar, 2010, pp. 586–589.

[11] H. Estrada, A. M. Rebollar, O. Pastor, J. Mylopoulos, An empirical evaluation of the i* framework in a model-based software generation environment, in: Advanced Information Systems Engineering, Springer, 2006, pp. 513–527.

[12] C. Martinie, P. Palanque, M. Winckler, Structuring and composition mechanisms to address scalability issues in task models, in: Human-Computer Interaction–INTERACT 2011, Springer, 2011, pp. 589–609.

[13] F. R. S. Paim, J. F. B. de Castro, Dwarf: An approach for requirements definition and management of data warehouse systems, in: Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International, IEEE, 2003, pp. 75–84.

[14] P. Giorgini, S. Rizzi, M. Garzetti, Grand: A goal-oriented approach to requirement analysis in data warehouses, Decision Support Systems 45 (1) (2008) 4–21.

[15] N. Prakash, A. Gosain, An approach to engineering the requirements of data warehouses, Requirements Engineering 13 (1) (2008) 49–72.

[16] S. Luján-Mora, J. Trujillo, I.-Y. Song, A UML profile for multidimensional modeling in data warehouses, DKE 59 (3) (2006) 725–769.

[17] J.-N. Mazón, J. Trujillo, An MDA approach for the development of data warehouses, DSS 45 (1) (2008) 41–58.

[18] E. Kavakli, Goal-oriented requirements engineering: A unifying framework, Requirements Engineering 6 (4) (2002) 237–251.

[19] P. Giorgini, S. Rizzi, M. Garzetti, Goal-oriented requirement analysis for data warehouse design, in: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP, ACM, 2005, pp. 47–56.

[20] M. Laitinen, M. Fayad, R. P. Ward, The problem with scalability, Communications of the ACM 43 (9) (2000) 105–107.

[21] M. Serrano, J. Trujillo, C. Calero, M. Piattini, Metrics for data warehouse conceptual models understandability, Information and Software Technology 49 (8) (2007) 851–870.

[22] X. Franch, The i framework: The way ahead, in: Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on, IEEE, 2012, pp. 1–3.

[23] M. Gorawski, M. Gorawski, Modified r-mvb tree and btv algorithm used in a distributed spatio-temporal data warehouse, in: Parallel Processing and Applied Mathematics, Springer, 2008, pp. 199–208.

[24] M. Gorawski, Extended cascaded star schema and ecolap operations for spatial data warehouse, in: Intelligent Data Engineering and Automated Learning-IDEAL 2009, Springer, 2009, pp. 251–259.

[25] M. Gorawski, R. Malczok, Indexing spatial objects in stream data warehouse, in: Advances in Intelligent Information and Database Systems, Springer, 2010, pp. 53–65.

[26] M. Gorawski, P. Jureczek, Regions of interest in trajectory data warehouse, in: Intelligent Information and Database Systems, Springer, 2010, pp. 74–81.

6

Aligning Data Warehouses with the Corporate Strategy

The content of this chapter corresponds with the following papers:
Maté, A. *and Trujillo, J. and Yu, E. Aligning Data Warehouse Requirements with Business Goals. Proceedings of the Sixth International i* Workshop (iStar 2013), CEUR Vol. 978, pp. 67-72. 2013 Valencia, Spain.*

The previous chapters in this PhD Thesis have focused on dealing with the pitfalls in data warehouse development overlooked by existing development approaches. In this chapter we go a step further than the current state of the art and tackle one of the main problems behind the high rates of failure in data warehouse development: the mismatch between IT specialists and business people [4]. Current data warehouse development proposals start by elicitating requirements from users in order to identify which data should be stored in the data warehouse. However, previous studies have pointed out that it is not feasible to extract a comprehensive and accurate set of requirements from decision makers, since each decision maker can only provide a partial, personal, point of view [41]. Therefore, elicitated requirements are not validated against the business strategy, thus it is possible that some of the required information is missing while some of the information stored is not aligned with business goals.

In turn, important business goals may be overlooked, and decision makers have to make an effort in order to interpret these data in business terms and take advantage of it. Nevertheless, these efforts are rarely successful [9], since business people may not have any IT knowledge required.

In order to avoid these problems, in the fourth part of this PhD Thesis, we align the data warehouse with the business strategy, thus ensuring that the data warehouse will support the business goals and identifying any potential business goals overlooked. Furthermore, this approach provides awareness of the context within the business strategy where each individual decision is being taken.

Aligning Data Warehouse Requirements with Business Goals

Alejandro Maté[1], Juan Trujillo[1], Eric Yu[2]

[1] Lucentia Research Group
Department of Software and Computing Systems
University of Alicante
{amate,jtrujillo}@dlsi.ua.es
[2] Department of Computer Science
University of Toronto
{eric}@cs.toronto.edu

Abstract. According to the Gartner Group, over 70% of Business Intelligence (BI) projects fail. Among the reasons are the different languages employed by IT and business people and the necessity of a long-term BI plan describing the goals of the organization. In current practice, when building the data warehouse (DW), strategic plans are rarely considered. In this paper, we propose a method to align the business plan with DW requirements analysis. By aligning the DW, we (i) validate the correctness of each decision makers' goals, (ii) ensure that their decisions and the DW contribute towards organization goals, and (iii) provide a long-term unified BI strategy. We instantiate this alignment by combining i* for DWs with strategic business models.

Keywords: Business Intelligence, business plan, data warehouses, alignment, BIM

1 Introduction

Data Warehouses (DW) integrate numerous heterogeneous data sources in multidimensional structures (i.e. facts and dimensions) in support of the decision-making process. In order to be successful, DW design processes include a requirement analysis step whenever the complexity of the data sources is high. However, despite the inclusion of this step, the different views between stakeholders and the different languages employed by IT specialists and business people are still one of the reasons why over 70% of Business Intelligence (BI) projects fail, according to the Gartner Group.

In order to minimize the probability of failure, recent DW development approaches introduce the use of goal models [4] during the requirements analysis step. These goal models are created through interviews and questionnaires among others. However, all these techniques rely on individual decision makers as the main source of knowledge. In most organizations where several departments work together, each decision maker has been proven to provide only a partial

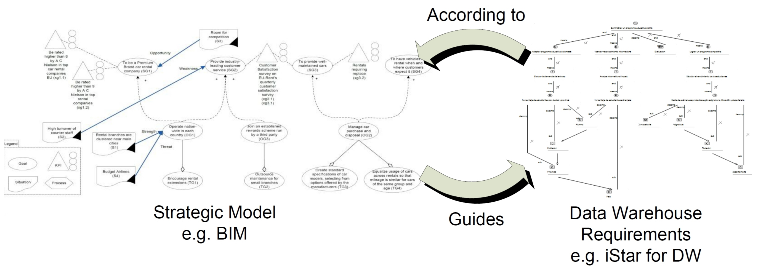


Fig. 1. Relationships between the strategic plan and DW requirements

view of the problem. Moreover, these partial views may not always be aligned with the business plan, which together with the gap between IT and business people makes it difficult to validate the goal models. In turn, the lack of alignment between individual decision makers and the business plan is translated into a lack of long-term enterprise Business Intelligence (BI) strategy, which is one of the key factors to successfully apply BI.

In this paper, we propose to tackle this problem by aligning the business plan with current BI and DW goal models. Therefore, we ensure that our BI-enabled decision making is consistent with the business plan. First, we elaborate a strategic goal model from the business plan to formalize business goals and trace business indicators. Then, we align and relate decision maker's goals with business goals. Therefore, we ensure that all the decision makers are contributing towards the overall goals of the enterprise, and identify which business goals are being overlooked by decision makers.

The remainder of this paper is structured as follows: Section 2 presents the main objectives of the research. Section 3 describes the alignment process and the main contributions of our work. Section 4 presents the conclusions. Section 5 describes the ongoing and future work in this area.

2 Objectives of the Research

The main objective of the research is to minimize the failure rate of DW and BI projects derived fom the misalignment between (i) different stakeholders due to different particular views and (ii) between stakeholders and IT specialists implementing the DW due to the different language employed by each of them. In order to achieve this objective, our goal is to provide (i) a method for aligning each particular decision maker view and the implementation of the DW with the overall business strategy and (ii) a set of models that can be used to instantiate the method and apply the approach.

3 Scientific Contributions & Tool

In this section we present the main contributions of our work, as well as the current tool support for the approach. Our contributions can be summarized as follows: (i) a method to perform the alignment between DW requirements and business goals, (ii) a set of mappings to instantiate the alignment method and correctly apply it for aligning i* for DWs with the BIM strategic model, and (iii) show how these mappings and models can be implemented in a tool in order to support our approach.

First, the alignment process starts by modeling decision makers' goals by means of techniques used in current DW approaches. For each decision maker, we create an strategic rationale i* as described in [4].

In parallel, or after DW requirements have been obtained, the business strategy model is created by using the information from the business plan. Business plans include information regarding business goals, the objectives associated with each goal, and may also include information regarding business processes. In order to create the business strategy model, we employ the Business Intelligence Model [1, 3], which includes four core concepts that allow us to formalize the elements described in the business plan: Goals, Situations, Indicators, and Processes (see [3]).

Once both models have been obtained, we proceed to perform the alignment process. First, we align the concepts between each metamodel, in order to ensure that we correctly relate each decision maker goal to the corresponding business goal. Then, in collaboration with a domain expert, we map each concrete decision maker goal to a specific business goal. Finally, we analyze the alignment and perform changes as necessary. The process described can be seen in Figure 2.

As seen in Figure 2, the alignment step takes as input (i) the particular goals of each decision maker, captured in the DW requirements model and (ii) the business goals, captured in the business stratey model. Each of these models can be instantiated according to different frameworks, since currently there is no standard for either of them, thus leading to different results. Therefore, in

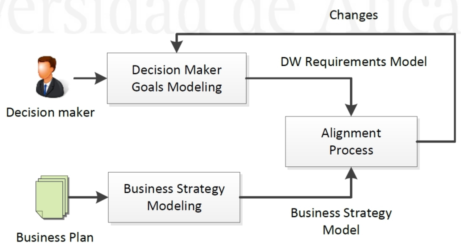


Fig. 2. Steps involved in the alignment process

order to apply the method, it is necessary to perform an ontological mapping between the concepts of the specific metamodels being used.

In our case, as previously described, we make use of i* for DWs in order to model DW requirements, and BIM for modeling the business strategy. Therefore, we analyze the definitions and characteristics of each of the elements included in the metamodels and align them to ensure their correct use. This alignment can be seen in Table 1.

In the group of aligned elements, we have the Strategic Goal and the Measure. First, a Strategic Goal in DWs is a long-term goal focused towards improving process results. In a similar way, Strategic Goals in BIM are long-term goals designed towards improving the results of the organization, and include KPIs to evaluate if the expected results are achieved. In order to guarantee that all the Strategic Goals pursued by decision makers are contributing towards the overall performance of the enterprise, all the Strategic Goals in DWs should be developed within the boundaries of Strategic Goals in the strategic model. In addition, in order to keep better track of the contributions of each decision maker's Strategic Goal, we can strengthen the condition and force each goal to contribute to, at most, a single business Strategic Goal. In other words, we state that the goals for decision maker i are *valid iff* given $D_i = \{\textit{Strategic Goals for decision maker } i\}$ and the set $BS = \{\textit{Strategic Goals in the strategic model}\}$, we have that $\forall S_j \in D_i, \exists! B_k \in BS : \textit{aligned}(S_j, B_k)$.

Table 1. Alignment between elements in I-Star for DW and elements in BIM

iStar	BIM	Details iStar	Details BIM
Business Process	Business Process	Not detailed, Strategies improve its results	May be detailed, Realizes goals
Strategic Goal	Strategic Goal	Future, Focused, Long-term, Qualitative or Quantitative, Improves process results	Future, Focused, Long-term Qualitative, Measured by KPI
Decision Goal	-	Future, Supports a Strategic Goal by taking decisions	BIM does not include a decisional aspect
Information Goal	-	Gathers information information to achieve a Decision Goal	BIM does not include a decisional aspect
Information Requirement	-	Describes information required to achieve an information goal	No counterpart
Context	-	Information unit	No counterpart
Measure	Indicator	Measures the performance of a Business Process without target values	Future, Time-Targeted, Long or Short-term, Measures Goal performance, Business Processes and Situations

If this condition is not satisfied, it would mean that one or more of the Strategic Goals pursued by the decision maker are not contributing towards the success of the overall strategic plan. Conversely, in order to guarantee that all the Strategic Goals in the strategic plan are being considered, at least one or more decision maker's goals, from one or more decision makers, should be aligned with them. Formally defined, we state that the strategic plan is being *supported* by the DW if $\forall B_k \in BS, \exists S_j \in D_i : \, aligned(S_j, B_k)$.

Finally, in order to guarantee the *consistency* of the alignment between both models, whenever a decision maker's strategic goal is improving a Business Process, the same Business Process should be related, at least indirectly, to the business goal aligned with the strategic goal. Formally, $S_j, BP_n \in D_i$, $B_k \in BS : related(S_j, BP_n) \wedge aligned(S_j, B_k) \Rightarrow \exists BP_m \in BS : related(B_k, BP_m) \wedge BP_n = BP_m$. Since not all the strategic plans may provide such a fine level of detail as to include Business Processes, we relax this condition and enforce it *iff* the Business Process appears in the strategic model.

In second place, we have Measures. Measures lack a target value, a threshold value, they do not have a time target, and they measure facts instead of goals. However, if these values are added to a measure, it can be converted into an Indicator for the organization.

Finally, in the group of non-matched elements, we have the Decision and Information Goals, the Information Requirements, and the Contexts. All of these elements are specifically related to the decision process. Thus, as there is no construct in BIM which has the appropriate semantics to capture the meaning of these elements, they cannot be aligned directly to the strategic model.

By analyzing the *support* and *consistency* of the alignment between particular decision makers' goals and business goals, we can validate DW requirements and identify potential changes that need to be performed to the business strategy.

Our approach is supported by our tool, the Lucentia BI suite, which allows us to model (i) user requirements by means of i* for DWs, (ii) business strategies by means of a particular implementation of BIM, and (iii) a trace metamodel that

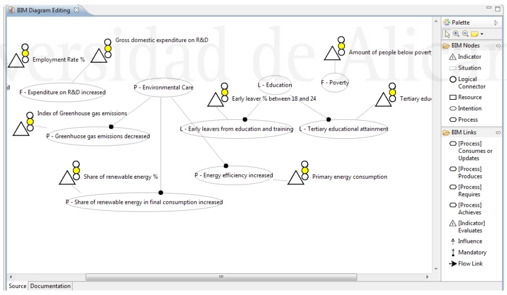

BIM Diagram Editing
Gross domestic expenditure on R&D
Employment Rate %
Amount of people below povert
F - Expenditure on R&D increased
P - Environmental Care
L - Education
F - Poverty
Early leaver % between 18 and 24
Tertiary edu
Index of Greenhouse gas emissions
P - Greenhuose gas emissions decreased
L - Early leavers from education and training
L - Tertiary educational attainment
Share of renewable energy %
P - Energy efficiency increased
Primary energy consumption
P - Share of renewable energy in final consumption increased
Palette
BIM Nodes
Indicator
Situation
Logical Connector
Resource
Intention
Process
BIM Links
[Process] Consumes or Updates
[Process] Produces
[Process] Requires
[Process] Achieves
[Indicator] Evaluates
Influence
Mandatory
Flow Link
Source
Documentation

Fig. 3. Business strategy editing using the Lucentia BI Tool

allows us to relate elements within different models, and has been succesfully integrated previously within a hybrid DW development approach [2]. The tool is based on Eclipse and includes a set of modules, each designed to support an specific task. An screenshot of the tool can be seen in Figure 3.

4 Conclusions

In this paper we have presented an alignment between the business strategy and i* DW requirements models. Our process results in an alignment that allows us to (i) validate DW requirements according to the business strategy and identify non-aligned goals, (ii) provide a long-term BI strategy to be pursued, including what information is being used by the organization to support each goal, (iii) identify the different decision makers participating in a business goal, thus providing awareness, and (iv) evaluate if decision makers are being successful by analyzing the values of indicators related to business goals.

While extending i* to consider all the elements within the business strategy would overcomplicate the model, our initial applications have shown that it can be combined with other models in order to capture both the particular viewpoint of each decision maker as well as the overall strategy of the organization.

5 Ongoing and Future work

The current ongoing work is focused on making it easier for businesses to apply our proposal. In order to achieve this, we plan to semi-automate the process of obtaining an strategic model directly from the business plan. Therefore, we are analyzing the viability of defining a series of pattern-based transformations in order to save time and costs. In addition, we are also extending the trace metamodel to capture the semantics described in the alignment method.

In the medium-term we plan to apply our approach to a real case study and evaluate the results. Since the business plan structure may vary from one organization to another, we plan to minimize the impact of this variability by mapping each particular plan to the standard Business Motivation Model, proposed by the Object Management Group group.

References

1. D. Barone, E. Yu, J. Won, L. Jiang, and J. Mylopoulos. Enterprise modeling for business intelligence. *The Practice of Enterprise Modeling*, pages 31–45, 2010.
2. A. Maté and J. Trujillo. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. *Information Systems*, 37(8):753 – 766, 2012.
3. A. Maté, J. Trujillo, and J. Mylopoulos. Conceptualizing and specifying key performance indicators in business strategy models. *ER'12*, pages 282–291, 2012.
4. J.N. Mazón, J. Pardillo, and J. Trujillo. A model-driven goal-oriented requirement engineering approach for data warehouses. *Advances in Conceptual Modeling–Foundations and Applications*, pages 255–264, 2007.

Part III

Appendix: Papers Already Submitted

7

Appendix A: Tracing Conceptual Models Evolution in Data Warehouses by using the Model Driven Architecture

The content of this chapter corresponds with the following papers:
Maté, A. *and Trujillo, J. Tracing Conceptual Models Evolution in Data Warehouses by using the Model Driven Architecture. Computer Standards & Interfaces. (2nd round of revision) Impact Factor: 0.978*

In this appendix we present an extension of our work for documenting the relationships between the data warehouse and the different data sources that is currently under second round of revision. This appendix includes a formalization of the basic trace types involved in the reconciliation process as well as an improved set of QVT relationships that allow to derive the final data warehouse schema from any trace configuration. Thus, we enable a quick reconfiguration and analysis of the data warehouse whenever a data source is changed or added.

Tracing Conceptual Models Evolution in Data Warehouses by using MDA

Alejandro Maté[a,*], Juan Trujillo[a]

[a]*Lucentia Research Group, Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n - 03690 San Vicente del Raspeig - Alicante, Spain*

Abstract

Developing a data warehouse is an ongoing task where new requirements are constantly being added. A widely accepted approach for developing data warehouses is the hybrid approach, where requirements and data sources must be accommodated to a new data warehouse model. During this process, relationships between conceptual elements obtained from user requirements and those supplied by data sources are lost, since no traceability mechanisms are included. Whenever (i) a new requirement is posed, (ii) an old requirement is reviewed, or (ii) a change is made in the data sources, the existing differences between data sources and the data warehouse schema make the designer waste additional time and resources trying to identify how it will be accomplished. Moreover, the missing information may create such a gap as to force the designer to perform the whole accommodation process again. Previously, we have defined a trace metamodel in order to trace user requirements to data warehouse conceptual models. In this paper, we propose an approach which preserves traceability at conceptual level for data warehouses. Therefore, we preserve existing relationships between elements, allowing us to easily identify how changes should be incorporated into the target DW, and derive it according to the new configuration. In order to minimize the effort required, we show how Query/View/Transformation rules can automate trace generation and provide a case study to show the applicability of the proposal.

Keywords: Data warehouses, traceability, conceptual models, user requirements, MDA, business intelligence

1. Introduction

Developing a data warehouse (DW) is an ongoing task where new requirements are constantly being added. Either as a result of the dynamic environment

*Corresponding author. Tel: +34 96 5909581 ext. 2737; fax: +34 96 5909326
Email addresses: `amate@dlsi.ua.es` (Alejandro Maté), `jtrujillo@dlsi.ua.es` (Juan Trujillo)

of the enterprise, or because new sources of information become available (i.e. social media), decision makers constantly pose new requirements and questions which need to be answered by analyzing information. This information is integrated from several heterogeneous sources and structured in the data warehouse in terms of facts and dimensions [1]. Therefore, the development of the DW is a continuous and complex process which must be carefully planned in order to meet user needs and incorporate new requirements. Thus, to develop the DW, three different approaches have been proposed: bottom-up or supply-driven, top-down or demand-driven, and hybrid [2, 3].

The first two approaches ignore at least one source of information until the end of the process, be either requirements or data sources. This lack of information leads to failure in some DW projects [2, 4] since they either ignore user needs or they assume that all the necessary data is available, which is not always the case. On the other hand, the hybrid approach makes use of both data sources and user requirements [3], solving incompatibilities by accommodating both requirements and data sources in a single conceptual model before implementing the DW. Nevertheless, the current accommodation process is performed much like a schema redesign process: successive modifications are made to the schema, removing, renaming, and adding new elements according to the designer's experience. In turn, the resulting DW schema may not match the data sources neither in structure nor in naming conventions, causing the existing traceability by name matching to be lost. Therefore, these correspondences must be identified again when (i) validating and reviewing old requirements, (ii) posing new requirements, or (iii) modifying data sources, all of which are error prone tasks. As a result, the costs in time and resources are increased while the quality of the final product is decreased [5].

In our previous works [6, 3, 7, 8], we defined a hybrid DW development approach in the context of the Model Driven Architecture (MDA) framework [9]. DW development presents a special peculiarity and idiosyncrasy which favors our approach. In DW development, data sources act as both a source of additional information as well as a limiting factor. In order to implement a requirement in the final DW, the required information must be present in the data sources, either directly or by deriving it. Therefore, we can clearly identify which is the desired structure of the DW (requirements), and which information is supplied (data sources). The final step in this process is adequately relating this information in such manner that changes can be easily traced and incorporated, instead of arbitrarily mixing the schema and making difficult further analysis tasks. Additionally, unlike in software development, the reconciling process can find elements which were not present in the requirements model (due to an oversight) but provide relevant information for decision makers [8]. Thus, it is interesting to trace elements present in the data sources which do not have a requirement counterpart, but are present in the final implementation of the DW, since they can point out to overlooked user requirements.

The automatic derivation is done by means of model to model transformations specified by Query/View/Transformation (QVT) [10] rules. QVT is a language defined by the Object Management Group (OMG) and proposed as

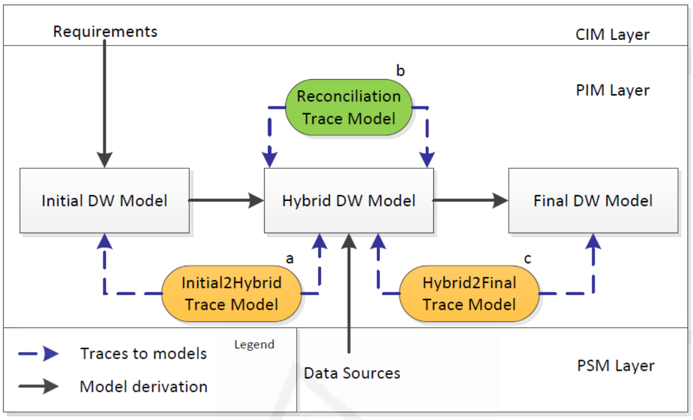


Figure 1: Overview of the approach

a standard to create model to model transformations. This language can be used to create both DW models as well as trace models. However, due to our experience, the reconciliation task can only be done manually or, at most, semi-automatically, since there is not enough information available to perform it fully automatically. The set of trace models employed in our approach are shown in Figure 1, and are further detailed in Section 3.

In the short version of this paper [11] we developed a set of traces for preserving the traceability of requirements at conceptual level. Now, in this extended version, we (i) provide a deeper review of the related work describing details of the existing traceability approaches, (ii) provide a formal definition of our traces, (iii) generalize a set of QVT transformations which allow us to derive the data warehouse from any trace configuration specified, and (iv) provide an extended case study which tests and shows better the application of the proposal.

The remainder of the paper is structured as follows. Section 2 presents related work about traceability and DWs. Section 3 introduces the necessary trace semantics in order to include traceability at the conceptual level in DWs. Section 4 presents the QVT rules for automatic derivation of traces. Section 5 presents an example of application, in order to show the benefits of our proposal. Finally, Section 6 outlines the conclusions and further work to be done.

2. Related Work

In this section, we will discuss the existing traceability research, its benefits and problems, and its current status in the DW field. Traditionally, traceability is focused on requirements. Either coming from the traditional RE [12, 13, 14, 15] or following a MDD approach [16, 17, 18], requirements are traced to their lower abstraction level counterparts. Therefore, traceability helps assessing the impact of changes in requirements and rationale comprehension, by identifying which parts of the implementation belong to each requirement [19], as well as reusability and maintainability [13]. However, the effort required due to the lack of standardization makes it difficult to apply traceability to projects, since even the basic concepts differ from author to author [16, 18]. Therefore, there is a special interest on automating traces and providing a framework with a set of concepts which can be extended.

Therefore, some recent works try to provide some degree of automation for recording the relationships between elements by following two different approaches. The first one, is to generate traces from already existing information. An advanced example is presented in [20], where the authors combine topic modeling with prospective traceability, creating traces as the user interacts with the different artifacts. The second approach is to make use of the logic behind automatic transformations. In this second approach, traces are created as the transformation logic generates a new version of a model, creating a trace model in addition to the target model. The trace model stores the relationships between elements in the source model and elements in the target model, and can be analyzed in an automated way by means of algorithms which take into account the different semantics. The former approach can be applied whenever a user interacts with an artifact, minimizing the necessity of manually adding traces. The latter can only be applied when models are automatically transformed. However, whereas the first approach may generate some incorrect traces, the second solution is not based on any assumptions or patterns, but rather in transformation logic, thus being less error prone.

Nevertheless, tracing the counterparts of a requirement at conceptual level, is not always straight-forward, even when following a MDD approach and exploiting transformation logic. Elements are refined by the developer before being transformed into the next model, altering the characteristics such as their name or even their structure. This process is repeated until the final implementation is achieved. Therefore, in order to maintain traceability between models, the result of these operations must still be traceable.

In DW development, the different steps can be clearly identified as the DW schema evolves through several conceptual models. Thus, in order to validate requirements and support incremental changes, we require to trace the representation of an element from one model to its counterpart in the next model.

In order to tackle this problem, different works from the Requirements Engineering (RE) [12, 13] and the Model Driven Development (MDD) communities [17, 18] have included traceability in software development processes. However, aside from our previous contribution in [21], where we defined a trace metamodel for tracking requirements to their corresponding conceptual elements, the traceability aspect has been practically overlooked in DW development. Some works mention the existence of certain mappings [2, 22], but they are not formally introduced nor include any specific semantic. Others rely on name matching as a means of implicit traceability, which cannot be applied if naming conventions and structures differ [3].

Our approach, presented in [3], applies MDD and is able to generate traces by exploiting transformation logic. While our approach applies a specific framework for DW development (MDA [9]), other development approaches [2, 22] make use of very similar layers, each one making use of his own conceptual representation [23, 24, 6] for modeling DWs. By generating traces between conceptual models, we enable to trace the different versions of an element, providing support for requirements validation, impact change and automated analysis, while minimizing the existing drawbacks in traceability.

Furthermore, by using MDA, we cut development time, since transformations from the top layer to the final implementation are performed in a semi-automatic way. In our approach, requirements are specified in a Computation Independent Model (CIM). Then, they are automatically derived into a conceptual model [6] at the Platform Independent Model (PIM) layer. The initial PIM model records the conceptual elements coming from requirements. In order to keep this model clean for further analysis, a hybrid PIM is derived by including information coming from the data sources, which is obtained by means of reverse engineering [25]. Afterwards, desired elements are marked and derived into the final PIM, which conceptually represents the implementation of the DW. This way, the hybrid model can be marked with multiple configurations, allowing us to derive alternative implementations of the DW.

Thus, in order to maintain accurate information about the implications of each requirement and data source in the DW schema, all elements must be traced along the successive refinements performed at the PIM layer. Therefore, traces must maintain the semantics of their relationships, allowing us to support automated analysis.

3. Traceability through Conceptual Models for Data Warehouses

As previously stated, in order to adequately perform incremental changes and analysis tasks, we require to trace information from both user requirements and data sources up to the final conceptual model. In order to achieve this goal, it is necessary to trace every concept as it evolves through the different conceptual models, shown in Figure 1. First, we will introduce our trace metamodel providing the basic concepts used for tracing elements along the PIM models. Then, we will motivate and introduce the relationships and traces in the hybrid PIM model, which are specially relevant to connect elements obtained from user requirements with elements derived from with data sources.

3.1. The Trace Metamodel

In order to trace conceptual elements up to the final PIM we require to include different semantics. This way, we can differentiate the relationships between elements and support further automatic operations. These semantics are included in the trace metamodel presented in Figure 2. The trace metamodel is an extension from the Atlas Model Weaver (AMW) proposal [26], and includes the necessary concepts in order to record both the semantic of the relationship as well as the context data (which transformation rule generated the trace or which elements are connected). The different semantics included cover the different steps in a DW development process:

First, *Satisfiability* links capture the relationships between elements in the user requirements model and conceptual elements, and are fully described in [21]. These links allow us to trace each requirement to its multidimensional counterparts, identifying which elements support multiple requirements and which elements are affected by the evolution of user requirements.

Second, *Derived_from* links capture the relationships between elements in the data sources and conceptual elements. These links are modeled following a similar approach as in [21] but focusing instead on the logic of the reverse engineering process [25]. As in the case of *Satisfiability* links, these links are out of the scope of this paper.

Third, the set of *Evolution*, *Overlap*, *Conflict* and *Rationalization*, are employed to trace elements as they evolve through the PIM layer. These semantic links are the focus of this paper:

- **Evolution** links are included to handle horizontal traceability which takes care of element changes in the same layer. These links track the different versions of an element at each PIM model. Evolution links serve as a way to navigate between models, allowing us to identify the refinements performed to attributes, dimension levels, and dimensions through the different PIM models. For example, a dimension level obtained from requirements may have its name changed and be enriched with attributes coming from existing reports. In addition to changing the name of an element, the developer can perform operations such as split or fuse to alter the initial result of a transformation while maintaining traceability. Each element traced by an *Evolution* link is considered to represent the very same concept in different stages of the development process.

- **Overlap** and **Conflict** links relate elements obtained from requirements with those coming from data sources. There are three main different possibilities depending on the situation when performing the reconciliation. First, the necessary information for an element obtained from requirements may not be available. In this case this element will not be related to any other element. Second, the necessary information may be available, requiring only to perform a cleaning process to use it. In this case, the element will be related to those which supply the information by means of *Overlap* links. Last, the necessary information may be available, but its structure may not be adequate. It may require to perform an integration process between different attributes or, in general, a transformation to obtain the required representation of the data. In this case, the element will be related to those which supply the information by means of *Conflict* links. These links are crucial for enabling traceability support, as they al-

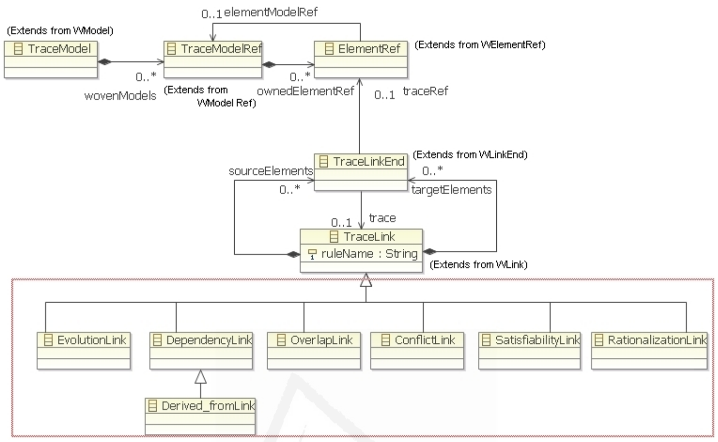


Figure 2: AMW Metamodel for traceability extended with semantic links for DWs

low us to merge elements coming from data sources and requirements and record the semantic of their relationship.

- **Rationalization** links are included as means of enabling the user to record his own annotations in the trace model regarding decisions, as well as provide reconciliated solutions for existing conflicts.

3.2. Trace Models in Data Warehouses

The previously defined trace types are recorded in different trace models included in our proposal, as shown in Figure 1. In our proposal, first we derive a PIM model from user requirements. This initial PIM is then refined with the necessary additions, such as attributes found in existing reports. Then, it is derived into a hybrid PIM, which includes conceptual elements from both user requirements and data sources. These elements are reconciled by means of traces. Finally, the developer selects those elements which wants to include in the target DW and derives the final PIM.

The first trace model, “a”, shown in figure 1, connects the initial PIM to the hybrid PIM in a pretty straight-forward way by means of *Evolution* traces. The initial PIM is maintained as a clean source model, for transformations combining elements from data sources and elements from requirements. This model is included in order to support automatic operations which require to track information related to requirements.

After we have derived the initial PIM, we proceed to create the hybrid PIM. First, we obtain a Platform Specific Model (PSM) from the data sources by means of reverse engineering. Then, elements in this PSM model are transformed into conceptual elements at the PIM level [25]. The result is a hybrid PIM, characterized by capturing different versions of the same concepts simultaneously. On the one hand, elements coming from the initial PIM present the structure of the DW according to users' expectations. On the other hand, elements transformed from the PSM model present the data "as-is" in the data sources. After obtaining the elements, the developer reconciles both versions by means of traces recorded in trace model "b" (see Section 3.3). Then, the developer marks which elements he wants to include in the final PIM.

The derivation process generates a set of evolution traces which are recorded in trace model "c". These evolution traces record which elements were chosen by the developer to be part of the final PIM, and allow to trace back to requirements and data sources the elements included in the DW. The traces generated differ depending on the relationship specified in trace model "b". Depending on the semantic of the relationships, the derivation process merges different elements into the same entity. Therefore, some of the traces in trace model "c" include multiple sources, and link elements from different entities. Our traces allows us to easily identify elements in the DW affected by a change in the data sources, elements related to each user requirement, or elements which did not obtain their information from data sources, for example because it is expected to retrieve it external sources.

After having defined the trace models which record the evolution of conceptual elements at PIM level, we will describe the reconciliation traces.

3.3. Reconciliation Traces at the Conceptual Level

Reconciliation traces represent special kinds of relationships which do not fit into the semantics of the metamodel being instantiated. Instead, these traces capture the relationships between elements from requirements and those from data sources at conceptual level. In our case, these relationships are defined by the *Overlap*, *Conflict*, and, if necessary, by *Rationalization* links.

As previously described, elements included in the hybrid PIM can be obtained from three different sources: (i) user requirements, (ii) data sources, and (iii) created by the developer to solve existing conflicts or create derived elements. In order to inter-relate these elements, the developer must manually identify which elements coming from user requirements match with each element coming from the data sources. Once identified, he captures the semantic of their relationship with the corresponding traces. After the developer has captured all the necessary relationships, he can mark which elements he wants to derive into the final PIM. Any change performed afterwards will be therefore easily traceable up to the final PIM, allowing us to analyze which requirements or data sources are also impacted by the change.

Reconciliation traces must be added manually since typically there is no knowledge about which element derived from the data sources is the counterpart to an element obtained from user requirements. User requirements are described in business terms, while data sources use naming conventions from IT. Moreover, data sources are typically focused on transactional processes while user requirements are oriented to describe the decision making process. Therefore, elements may differ in name, attributes and even in levels present in the dimension hierarchy. This problem may be partially solved depending on the amount of information provided from user requirements and the data sources. An example of this approach applied in software engineering is [20], although it is out of the scope of this paper.

In order to adequately relate user requirements with data sources, we proceed to apply the specified trace links as follows:

- **Overlap** links are employed whenever an element obtained from user requirements is complementary with the data existing in the data sources. This relationship describes a situation where both elements obtain their values from the same domain and their structure is equivalent. Thus, both elements could be interchanged without altering the meaning of the schema. Therefore, the final concept is conceived as the fusion between the elements provided by the data sources with those specified by user requirements, renaming them accordingly to the user needs. Formally defined, given two elements $e_1 \in D_1, e_2 \in D_2 \rightarrow D_1 \cap D_2 \neq \emptyset$, where D_1 is the source domain and D_2 is the target domain.

- **Conflict** links are employed whenever an element coming from user requirements is contrary to its representation coming from the data sources. This relationship describes a situation where both the element coming from user requirements and its representation from data sources refer to the same concept, but their domain is different. Therefore, one concept can not be interchanged with the other without obtaining a different schema, including different ways to aggregate the data. For example, a "Customer" level coming from user requirements includes as attributes "name" and "surname" but data sources only present a single attribute for representing this information "full name". These attributes are conflicting since their domains are different unless a transformation is applied. Moreover, if these attributes constitute the *Descriptor* (identifier) of the level, the resulting schema would present different aggregations depending if we used "name" or "full name". Formally defined, given two (or more) elements $e_1 \in D_1, e_2 \in D_2 \rightarrow D_1 \cap D_2 = \emptyset$, where D_1 is the source domain and D_2 is the target domain.

  In order to solve this situation, two solutions can be applied. The first solution is to choose one representation as correct, and derive the corresponding elements. In the previous example "name", "surname", and "full name" would be related by means of a *Conflict* link. After creating the link, we would mark the desired elements, which would be derived into the final PIM. The second solution is to provide one or more reconciliating elements by means of *Rationalization* links.

- **Rationalization** links are employed whenever the developer requires to create a new element which reconciliates an existing conflict. Using the previous example, the developer may create a new level, "Standardized-Customer", including "surname" and a new attribute "addressing name". This new attribute could include the "full name" information, as well as the way of addressing. Additionally, *Rationalization* links may also be applied to create derived conceptual elements. These links trace which elements contribute in the creation of a previously unexisting element.

After having explained the different semantic links included in the conceptual model refining process, we will present the necessary QVT rules for the automatic derivation and record of traces.

4. Automatic Derivation of Traceability Models in Data Warehouses

In this section, we will discuss the necessary transformations to automatically generate the traces between conceptual elements and store them in trace models, which can be updated over time.

According to our proposal for developing DWs [3, 7], we use a hybrid approach based on MDA. First, elements are derived in an initial PIM model from requirements. Then, we successively refine this initial PIM model and reconciliate it with data sources obtaining the hybrid PIM. Finally, we mark the desired elements and derive the final PIM. All these transformation are done by means of QVT rules. QVT rules specify a transformation by checking for a defined pattern in the source model. Once the pattern is found, a QVT rule transforms elements from the source metamodel into the target metamodel.

The process to derive the final PIM according is based on two general QVT rules. Both rules make use of two different metamodels: the conceptual DW metamodel [6] and the trace metamodel, presented in this paper. These two metamodels are used in four different models involved in these rules: the source DW model in the top-left corner, the target DW model in the top-right corner, the reconciliation trace model in the lower-left corner, and the trace model for evolution traces in the lower-right corner.

The first rule, *OverlapRule*, defines how overlapping elements are derived. This rule creates an *Evolution* link from the hybrid to the final PIM. At the top-left corner of Figure 3 we can find the conceptual elements from the hybrid PIM. This rule establishes that all the elements related by an overlap relationship, "E1" and "E2", act as sources of the transformation. At the top-right corner we can find the resulting target element, "R1". The values assigned to the properties of this element, i.e. its name, are the ones stored in the marked element. At the lower-left corner we can find the trace link relating "E1" and "E2". In this case, the trace link is an Overlap type link which has two trace link ends, corresponding to the elements linked. Finally, at the lower-right corner of the Figure we can find the new generated trace link, which is stored in trace model "c". This trace is an Evolution type link, containing "E1" and "E2" as sources, and "R1" as the target. In this way, the Evolution link allows us to automatically identify if the target element will be affected by a change in requirements or in the data sources.

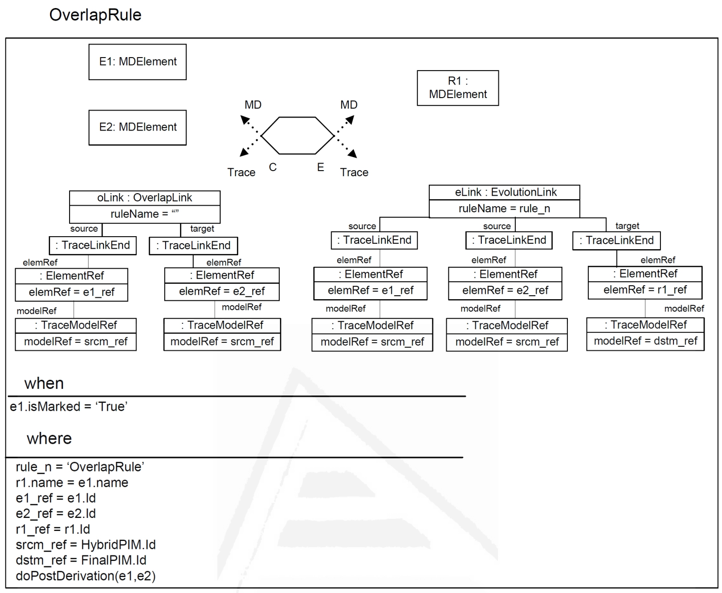


Figure 3: Generic QVT rule for deriving overlapping required elements and sources and creating their *Evolution* trace link

The “C” at the center of the figure means that the source models are checked to evaluate if the specified pattern exists. On the other hand, the “E” specifies that the target models are enforced. Thus, each time that the described pattern is found in the source models, the target patterns are generated in the resulting target models. The “When” clause establishes that this rule will apply only when an element is marked by the designer. Finally, the “Where” clause establishes other operations performed after the generation of “R1”. The most relevant operation in this clause is the function call “doPostDerivation”. This call performs actions such as analyzing the different attributes to be included in the case of levels and establish the roll-up hierarchies in case of dimensions.

The second rule, *ConflictRule*, defines how conflicting elements are derived. This generic rule has a significant difference with the previous one: only the marked element is considered as the one providing information for the new generated element. Therefore, this rule establishes that only “E1”, which was marked by the developer, acts as a source for the Evolution trace link.

Any element not marked by the designer is ignored in the derivation process. However, these elements are maintained for future analysis tasks.

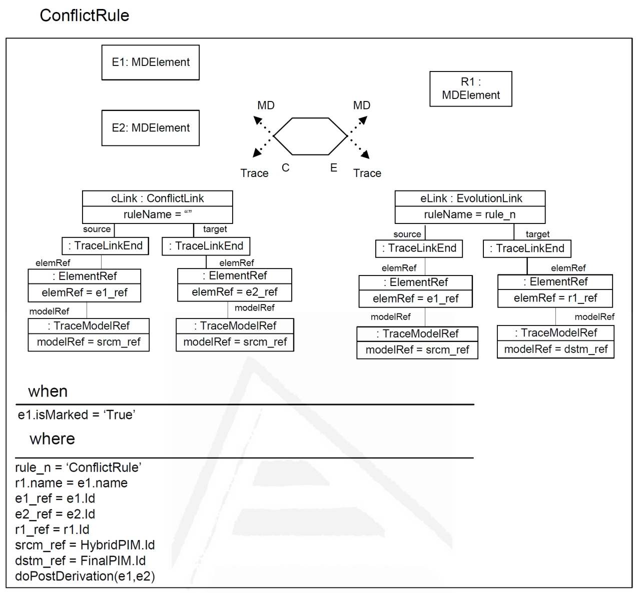


Figure 4: Generic QVT transformation for deriving a data warehouse element from conflicting required elements and sources

Both rules presented in this section allow us to obtain the final DW conceptual model while at the same time generating all the required traces in the process. In this way, the whole trace model from the hybrid to the final PIM is created in an automatic way and does not require user intervention.

5. Case Study

In this section, we will present a case study for our traceability proposal, showing how the traces can be used to derive the different configurations of the final PIM, which represents the DW implementation. This case study is inspired from a real world project with another university, and describes the basic process of our proposal while making it easier to read the data source model. All the diagrams are presented with our iconography for DWs [6], which presents UML classes stereotyped according to the multidimensional elements in DWs.

5.1. Creation of the Hybrid PIM and Intra-Model Trace Links

A university wishes to improve its educative process. In order to do so, a DW is designed to store the necessary information for the decision making process. The initial PIM, shown in Figure 5, is derived from user requirements and refined with the expected attributes. This PIM includes 4 dimensions and a single measure. First, we have the "Subject" dimension. A subject is expected to include its code, a name, the credits and a description of the subject. Additionally, subjects can be aggregated by their "Type". Next is the "Teacher" dimension. A teacher includes a code, a name and the years of experience he has. Furthermore, teachers can be aggregated according to their "Department", their "Faculty" or their job "Type". The third dimension is the "Student" dimension, which stores information regarding students. A student has a name and a code assigned by the university registry. Students can either be aggregated depending on their "Income" range or the "HoursofStudy" they spend each week. Finally, the "AcademicPeriod" dimension has a code and a name assigned to it. These academic periods can be aggregated into academic years. All these dimensions allow us to analyze a single performance measure which is the "Grade" obtained by the students.

As opposed to this initial PIM, the model created from the data sources[1] presents a higher number of attributes, different naming conventions, and fewer aggregation paths for the dimensions. The model created from the data sources

[1]The data sources model has been restricted to the most relevant concepts for the case study at hand, down from over a hundred tables

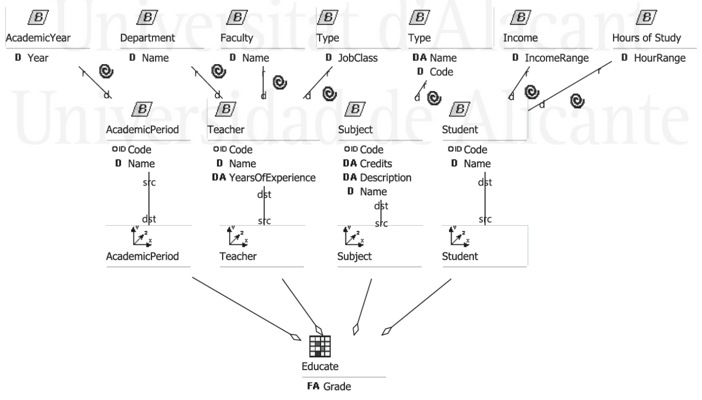


Figure 5: PIM model obtained from user requirements

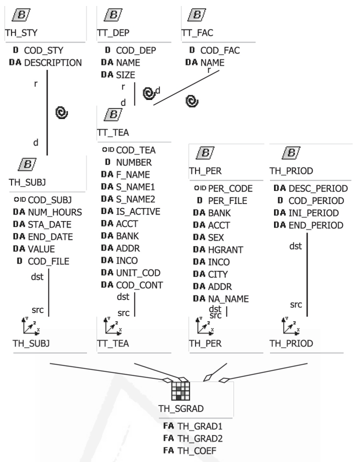


Figure 6: Datasources model obtained by reverse engineering

can bee seen in Figure 6. The first dimension is "TH_SUBJ", which would correspond to the previous "Subject" dimension. This dimension includes a code for the subject as we expected, the duration in hours of the subject, a starting date as well as an ending date, a value which cannot be easily identified, and a code for the file of the subject. The subjects may also be grouped by type, as expected, according to the data sources. The next dimension is "TT_TEA", containing the information related to teachers. The information recorded for a teacher includes his name and surname, a mark for indicating if he is active or not, his bank information, address, three different codes, and his income. According to the data sources, teachers can be grouped either by department or by faculty. If we wanted to group them by their job position, we would need additional elements. The third dimension present is "TH_PER", which stores information about the people registered in the university. The information stored includes a code for the person, his name, the number corresponding to its file, and other personal information similar to the case of teachers. According to the data sources, this level cannot be aggregated into any other. Finally, the academic periods are stored in "TH_PRIOD", which contains the description of the period, its code, the initial date and the final date of the period.

After describing both conceptual models, we can analyze the existing differences. First, there are differences in how levels are identified. For example, according to the model obtained teachers and subjects are identified by their number instead of by their name. This leads to different aggregated measures in the cube than initially expected since the same subject may have different codes

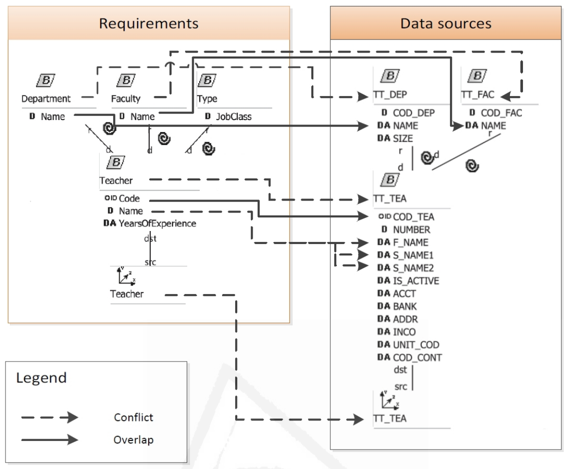


Figure 7: Traces between conceptual elements, Teacher dimension

when it is included in multiple academic plans. Moreover, some attributes are structured in a different way. For example, "Name" in "Teacher" level is actually fragmented in three different fields. Other attributes do not even appear, such as the subject name, which may be included in its description. Finally, some levels are missing, such as the "Type" of a teacher or the "AcademicYear".

All these differences can be explicitly captured by applying our approach. The process starts by relating attributes between levels, applying the definitions specified in Section 3.3. For each attribute in the requirements PIM model, we analyze which data source attributes provide the necessary data. Then, we apply the definitions accordingly to levels and dimensions. The result of this process is seen in Figure 7 for "Teacher" dimension, and in Figure 8 for "Subject" dimension, which are the most complex ones in this schema.

In this Figure we have related user requirements with data sources at conceptual level. As shown in the Figure, most of the levels specified are actually identified by different attributes in the data sources. Furthermore, its specially significant the case where the "Subject" level does not have any counterpart for its descriptor in the data sources, thus a correspondence between the expected set of subjects and the data provided cannot be established unless the schema is modified. Finally, we can see a few attributes in the data sources which are not expected but could be useful for the analysis, such as "IS_ACTIVE" determining if a teacher is still active or not.

The lack of key information regarding subjects forces to implement the DW with the data provided as-is until the missing information is provided.

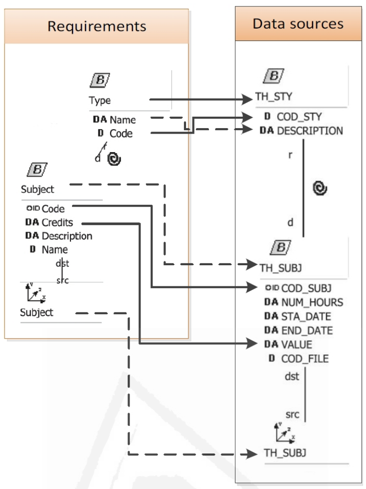


Figure 8: Traces between conceptual elements, Subject dimension

5.2. Derivation of the Final PIM

After we have obtained the hybrid PIM, and defined the necessary intra-model relationships, we can derive the final PIM. In order to obtain the final PIM, desired elements are marked to be included in the final PIM and then they are derived along with *Evolution* traces. Given the high number of traces present, performing this process manually would be time consuming and error prone. Therefore, our specified QVT rules allow us to avoid this pitfall by performing it automatically. The result is shown in Figure 9 where a handful Evolution traces are shown covering the three different possibilities: (i) evolution from an overlap, (ii) evolution from a conflict, and (iii) evolution from a single element. Finally, in Figure 10, the whole final PIM schema can be seen.

The final PIM can be further refined by renaming, or deleting elements which were initially included for the implementation but are lacking the necessary information to be filled. In addition, with our approach, we can perform quick updates as changes are introduced into user requirements or data sources.

After implementing the DW, the data sources are updated to include the subject name. To analyze how this change affects us, we analyze the hybrid PIM model and evaluate the different levels related to "TH_SUBJ". The process shows that level "Subject" was initially missing the necessary information to be identified, thus it could not be implemented properly. After the update, this information is no longer missing, and the new column can be related to the descriptor. Therefore, now we can properly derive "Subject" instead of

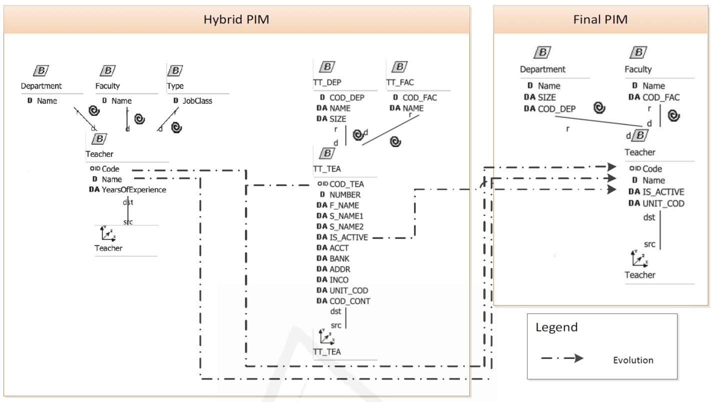


Figure 9: Evolution traces relating conceptual elements from hybrid PIM (left hand) with elements in the final PIM (right hand)

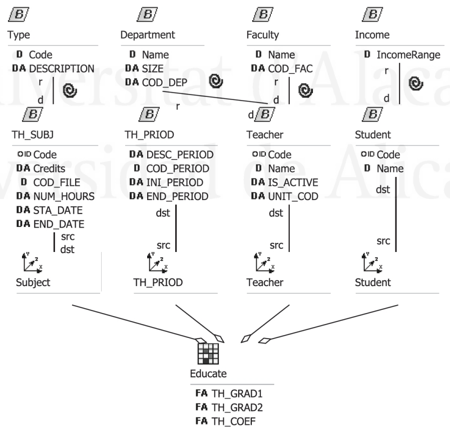


Figure 10: Final PIM representing the data warehouse implementation

"TH_SUBJ", thus being able to analyze subjects as users initially expected, instead of differentiating them by code and obtaining different measures.

With the previous development process, this change would have required to either (i) explicitly keep track of all the missing attributes, or (ii) perform the whole reconciliation process again, since the hybrid PIM was transformed, thus it has to be matched against the data sources after the update. Instead, by applying our proposal, new changes can be quickly identified and have a minor impact in the DW schema, thus they can be easily assessed and incorporated into the final DW.

6. Conclusions and Future Work

In this paper, we have proposed a traceability approach in order to explicitly specify the relationships between elements at the conceptual level in DWs. We have shown the necessary trace semantics to record these relationships and have proposed a set of guidelines in order to apply our proposal. Furthermore, we have shown how trace derivation and recording can be automated. We have also exemplified the application of the proposal by means of a case study with a university. The great benefit of our proposal is that the reconciliation task is only performed once per element and is preserved for further derivations or changes. Therefore, we avoid having to repeteadly inspect the data sources in order to match conceptual elements coming from requirements with those coming from data sources. In turn, we reduce the amount of time and resources spent and improve the maintainability of the system.

We are currently working on developing a set of algorithms to analyze the quality of the matching performed and automatically propagate changes. In the long term, we plan on analyzing the possibility of generating an initial subset of traces automatically, thus helping the developer to perform the reconciliation process.

Acknowledgments.
This work has been partially supported by the MESOLAP (TIN2010-14860) and SERENIDAD (PEII-11-0327-7035) projects from the Spanish Ministry of Education and the Junta de Comunidades de Castilla La Mancha respectively. Alejandro Maté is funded by the Generalitat Valenciana under an ACIF grant (ACIF/2010/298).

[1] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, B. Becker, The data warehouse lifecycle toolkit, Wiley, 2011.

[2] P. Giorgini, S. Rizzi, M. Garzetti, GRAnD: A goal-oriented approach to requirement analysis in data warehouses, Decision and Support Systems 45 (1) (2008) 4–21.

[3] J. Mazón, J. Trujillo, An MDA approach for the development of data warehouses, DSS 45 (1) (2008) 41–58.

[4] A. Bitterer, K. Schlegel, D. Laney, Predicts 2012: Business Intelligence Still Subject to Nontechnical Challenges (2011).
URL http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1873915

[5] S. Winkler, J. von Pilgrim, A survey of traceability in requirements engineering and model-driven development, Software and Systems Modeling 9 (2010) 529–565.

[6] S. Luján-Mora, J. Trujillo, I.-Y. Song, A UML profile for multidimensional modeling in data warehouses, Data & Knowledge Engineering 59 (3) (2006) 725–769.

[7] J.-N. Mazón, J. Pardillo, J. Trujillo, A model-driven goal-oriented requirement engineering approach for data warehouses, ER'07, Springer-Verlag, 2007, pp. 255–264.

[8] J.-N. Mazón, J. Trujillo, A hybrid model driven development framework for the multidimensional modeling of data warehouses, SIGMOD Record 38 (2) (2009) 12–17.

[9] OMG, A Proposal for an MDA Foundation Model.

[10] OMG, The Meta-Object Facility 2.0 Query/View/Transformation. Final Adopted Specification.

[11] A. Maté, J. Trujillo, Incorporating traceability in conceptual models for data warehouses by using MDA, Conceptual Modeling–ER 2011 (2011) 459–466.

[12] O. Gotel, S. Morris, Macro-level Traceability Via Media Transformations, in: Requirements Engineering: Foundation for Software Quality, Vol. 5025 of LNCS, Springer Berlin, 2008, pp. 129–134.

[13] B. Ramesh, M. Jarke, Toward reference models for requirements traceability, IEEE Transactions on Software Engineering 27 (1) (2001) 58–93.

[14] G. Spanoudakis, A. Zisman, Software traceability: a roadmap, Handbook of Software Engineering and Knowledge Engineering.

[15] Y. Yu, J. Jurjens, J. Mylopoulos, Traceability for the maintenance of secure software, in: IEEE International Conference on Software Maintenance, IEEE, 2008, pp. 297–306.

[16] N. Aizenbud-Reshef, B. Nolan, J. Rubin, Y. Shaham-Gafni, Model traceability, IBM Systems Journal 45 (3) (2006) 515–526.

[17] F. Jouault, Loosely coupled traceability for atl, in: ECMDA-TW, Nuremberg, Germany, 2005, pp. 29–37.

[18] R. Paige, G. Olsen, D. Kolovos, S. Zschaler, C. Power, Building model-driven engineering traceability classifications, ECMDA-TW (2008) 49–58.

[19] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, E. Merlo, Recovering traceability links between code and documentation, IEEE Transactions on Software Engineering 28 (10) (2002) 970–983.

[20] H. Asuncion, A. Asuncion, R. Taylor, Software traceability with topic modeling, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, ACM, 2010, pp. 95–104.

[21] A. Maté, J. Trujillo, A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses, Information Systems 37 (8) (2012) 753 – 766.

[22] P. Vassiliadis, Data Warehouse Modeling and Quality Issues, Ph.D. thesis, Athens (2000).

[23] A. Abelló, J. Samos, F. Saltor, YAM2: a multidimensional conceptual model extending UML, Information Systems 31 (6) (2006) 541–567.

[24] M. Golfarelli, D. Maio, S. Rizzi, The dimensional fact model: a conceptual model for data warehouses, International Journal of Cooperative Information Systems 7 (2) (1998) 215–247.

[25] J.-N. Mazón, J. Trujillo, J. Lechtenbörger, Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms, Data & Knowledge Engineering 63 (3) (2007) 725–751.

[26] M. Del Fabro, J. Bézivin, P. Valduriez, Weaving Models with the Eclipse AMW plugin, in: Eclipse Modeling Symposium, Eclipse Summit Europe 2006, Esslingen, Germany, 2006.