Computer Standards & Interfaces 36 (2014) 831–843

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

# Tracing conceptual models' evolution in data warehouses by using the model driven architecture

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

*Article history:*
Received 8 May 2013
Received in revised form 7 December 2013
Accepted 2 January 2014
Available online 11 January 2014

*Keywords:*
Data warehouses
Traceability
Conceptual models
Business intelligence
MDD
MDA
QVT

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 reconciliated data warehouse model. During this process, relationships between conceptual elements specified by user requirements and those supplied by the data sources are lost, since no traceability mechanisms are included. As a result, the designer wastes additional time and effort to update the data warehouse whenever user requirements or data sources change. In this paper, we propose an approach to preserve traceability at conceptual level for data warehouses. Our approach includes a set of traces and their formalization, in order to relate the multidimensional elements specified by user requirements with the concepts extracted from data sources. Therefore, we can easily identify how changes should be incorporated into the data warehouse, and derive it according to the new configuration. In order to minimize the effort required, we define a set of general Query/View/Transformation rules to automate the derivation of traces along with data warehouse elements. Finally, we describe a CASE tool that supports our approach and provide a detailed case study to show the applicability of the proposal.



## 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, or due to new sources of information becoming 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. Then, it is structured in terms of facts and dimensions in the DW [1]. Therefore, the development of the DW is a continuous and complex process that must be carefully planned in order to meet user needs and incorporate new requirements. To this aim, three different development approaches have been proposed: bottom-up or supply-driven, top-down or demand-driven, and hybrid [2,3].

The first two approaches ignore one source of information until the end of the process, either requirements or data sources. This lack of information leads to failure in some DW projects [2,4] since they either (i) ignore user needs or (ii) 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 neither match the data sources in structure nor in naming conventions. As a result, existing traceability by name matching is 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. Consequently, time and resources required 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]. The idiosyncrasy of DW development 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 the desired structure of the DW (requirements), and what information is supplied (data sources). The final step in this process is to adequately relate this information in order to easily trace and incorporate changes, instead of arbitrarily mixing the schema, and making it difficult to perform further analysis tasks. Additionally, unlike in software development, the reconciling process may find elements not present in the requirements model (due to an oversight) that provide relevant information for decision makers [8]. Thus, it is interesting to trace elements present in the data sources that do not have a requirement counterpart, but are present in the implementation of the DW, since they help to elicitate overlooked user requirements.

* 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).

0920-5489/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2014.01.004

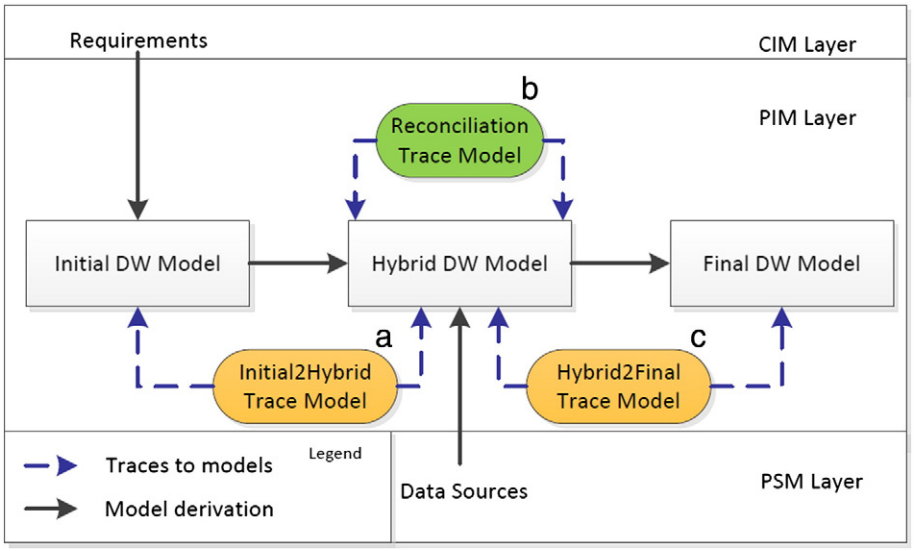


**Fig. 1.** Overview of the approach.

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 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 at most semi-automatically, since there is not enough information available to fully automate it. The set of trace models employed in our approach are shown in Fig. 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 the 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 this 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 a set of QVT rules for automatic derivation of traces. Section 5 presents a case study to show the applicability of our proposal. Finally, Section 6 outlines the conclusions and further work to be done.

## 2. Related work

In this section, we will discuss existing traceability research, its benefits and problems, and its current status in the DW field. Traditionally, traceability has been focused on requirements. Either coming from the traditional Requirements Engineering (RE) [12–15] or following a Model Driven Development (MDD) approach [16–18], requirements are traced to their lower abstraction level counterparts. Therefore, traceability helps assess the impact of changes in requirements and in rationale comprehension, by identifying which parts of the implementation belong to each requirement [19]. Additionally, it also improves reusability and maintainability [13]. However, 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 basic concepts that can be extended.

In order to provide some degree of automation, recent works record the relationships between elements by following two different approaches. First, generating traces from already existing information. An advanced example is presented in [20], where the authors combine topic modeling with prospective traceability as the user interacts with the system. Second, making use of the logic behind automatic transformations. In this second approach, the transformation logic generates a set of traces in addition to the new version of a model. Traces record the relationships between elements in the source and target models, and can be analyzed in by means of algorithms that take into account their 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 based on 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 their characteristics or even their structure. This process is repeated until the final version is obtained. Therefore, in order to maintain traceability between models, the result of these operations must be traceable.

In DW development, the different steps can be clearly identified as the DW schema evolves through several conceptual models. However, the differences in the language used by decision makers, and the language used by IT experts, make it difficult to reconcile data sources and the target DW schema. This communication problem is analyzed in [21], where the authors propose to tackle this problem by means of ontologies to improve the communication between parties. A combination of ontologies and traceability could help to produce a seamless integration of new data sources and changes into the DW. Thus, in order to validate requirements and support incremental changes, we require the tracing of 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 RE [12,13] and the MDD communities [17,18] have included traceability in software development processes. However, aside from our previous contribution in [22], where we defined a trace metamodel for tracing requirements to multidimensional structures, the traceability aspect has been overlooked in DW development. Some works mention the existence of mappings between the models involved in DW development