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

M. Jeusfeld, L. Delcambre, and T.W. Ling (Eds.): ER 2011, LNCS 6998, pp. 459–466, 2011.

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

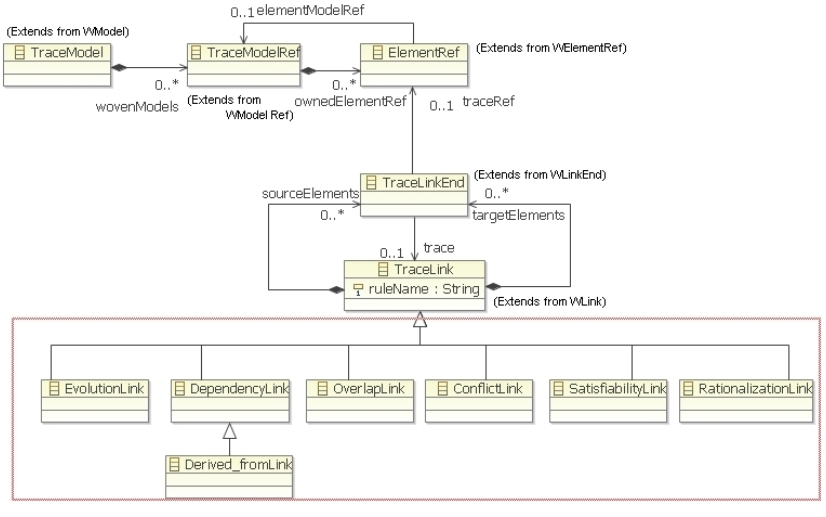

Fig. 1. Trace metamodel with semantic links for DWs

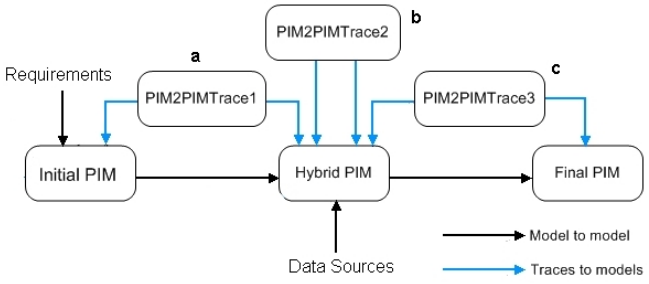

Fig. 2. Trace models linking the different PIM models in our DW approach

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

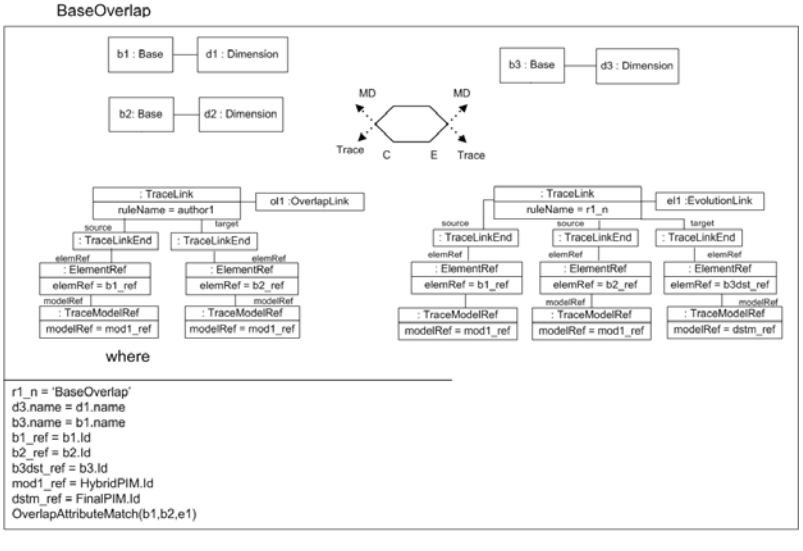

Fig. 3. QVT rule for deriving overlapping bases and creating their *Evolution* trace link

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,

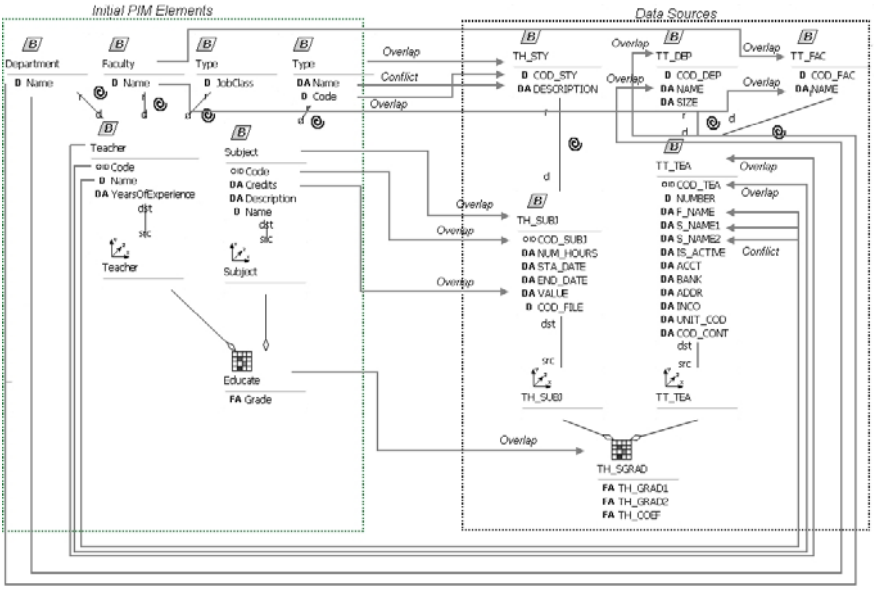

Fig. 4. Intra-model PIM traces relating conceptual elements from requirements (left) with elements from data sources (right)

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 MESO-LAP (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)