A Trace Metamodel Proposal Based on the Model Driven Architecture Framework for the Traceability of User Requirements in Data Warehouses

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 we lose the relationships between requirements, elements in the 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 Load processes. In this paper, we propose the first trace metamodel for DWs and focus on the relationships between requirements and conceptual models. We propose a set of Query/View/Transformation rules to include traceability in DWs in an automatic way, allowing us to trace every requirement to the conceptual model and further increasing user satisfaction.

Keywords: Data Warehouses, traceability, user requirements, 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 [10, 12]. 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, top-down, and hybrid), were proposed [21,4].

The first approach 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 [21] the DW fails to meet the user expectations. The second approach 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 [16]. 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 acommodating 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 approaches, 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. Nevertheless, in the hybrid approach, 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. Due to our experience, by following the hybrid approach, changes 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 [24, 26, 29], making 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 [1, 5, 11, 22, 2, 8, 9, 23, 26], it has been almost completely overlooked in DW development. To the best of our knowledge, the only references to requirements traceability in DW are those from [15], which only mention implicit traceability by name matching.

In our previous works [14, 15, 16, 17], we defined a hybrid DW development approach in the context of the Model Driven Architecture (MDA) framework [19]. DWs are sensitive to be developed by using MDA, cutting development time and making the process less error prone, since transformations from the top layer to the final implementation are performed in an semi-automatic way. In our approach, requirements are specified in a Computation Independent Model (CIM) by means of a UML profile [15] based on the i* framework [30]. 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 and Platform Specific Model (PSM) layer) until the final implementation is achieved, as seen in figure 1.

The automatic derivation is done by means of model to model transformations specified by Query/View/Transformation (QVT) [20] rules. QVT is a language defined by the Object Management Group (OMG) and proposed as a 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 and incurs in additional costs when requirements change.

In this paper, we complement our previous works with the inclusion of the first traceability metamodel for DWs and an automatic derivation of the correspoding trace models. In this way, by including traceability, we improve the reusability, maintainability and rationale comprehension of the models [24, 29], and we are

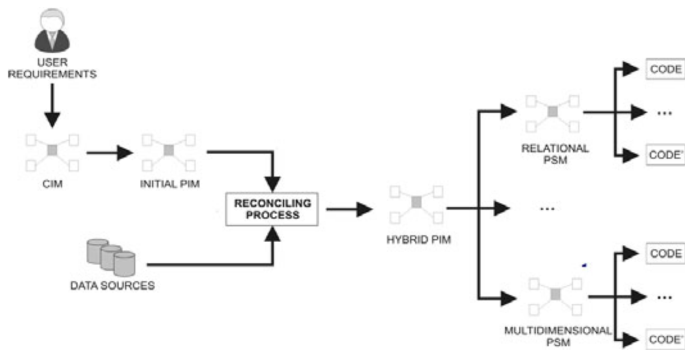

Fig. 1. Layers in our DW approach

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.

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. Section 5 presents an example of application, in order to show the benefits of our proposal and Section 6 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 have some common issues.

Most of the work done until now has been in the RE field [1,2,3,8,9,23,26,31]. Some authors [8, 29] 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, 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 [2, 23, 24]. Traceability helps assesing the impact of changes and rationale comprehension, by identifying which parts of the implementation belong to each requirement [2]. 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 [23, 24].

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 [24]. This situation creates a problem which makes difficult to succesfully apply traceability.

In order to alleviate the first drawback, a classification of eight categories for traces was presented in [26]. The second and third drawbacks can be solved by automating the trace recording. However, 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 difficult the comprehension and visualization of the trace model, which usually has a huge number of traces already [28].

On the other hand, in the MDD field, the MDA framework is used [1, 5, 11, 22, 28]. The automatic derivation process starts from a CIM layer, where the requirements are specified as models, usually by means of goal-oriented models [7, 13, 15, 18, 21, 25, 32]. In this sense, 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. There are mainly two definitions of traceability in the MDD community. The definition we will use in this paper comes from [22]; 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 [16] or a similar set of layers [27]. Currently, whenever a change to an element is done, the traceability as defined in [22] 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 [27] could be provided in a more automated way with traceability support, as opposed to performing the process manually, allowing us to increase the quality of the final implementation.

Due to the peculiarity and idiosyncrasy of data warehouses, we will need to difference 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 [14].

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 semantic 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 [19].

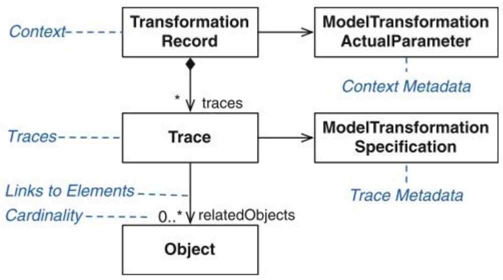

Fig. 2. Metamodel for traceability in MDA

The metamodel, presented in figure 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 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, which was the rule of the transformation created each trace.

According to this proposal there is a core metamodel for the ATLAS Model Weaver (AMW) [6], 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 figure 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

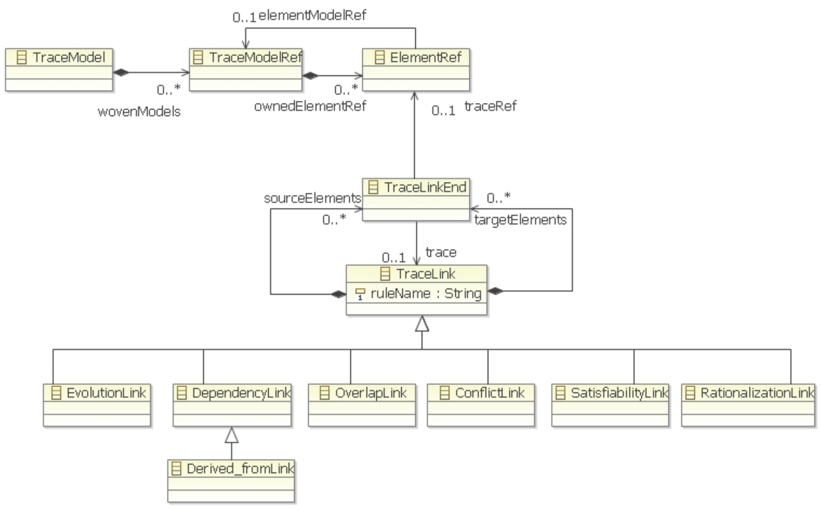

Fig. 3. AMW Metamodel for traceability extended with semantic links for DWs

result of the automatic derivation. A trace link can also have one parent, as well as a set of children trace links. This is an important feature, since it allows us to group traces forming hierarchies, providing different levels of detail in the trace models. Therefore, the trace models allow us to visualize over a hundred traces, which they typically store, in a scalable way. 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 [26]. We could use a reduced set of links, since in our case *Overlap* and *Conflict* are very similar. However, for the sake of standarization, we include the whole set. 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.

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 [15, 16], we will introduce the trace models shown in figure 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 [15, 16]) 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 sense, 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

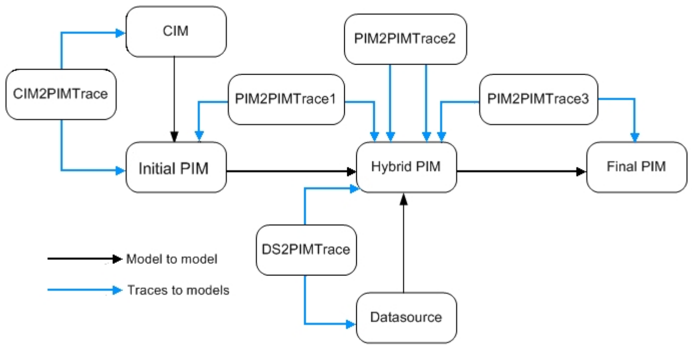

Fig. 4. Inclusion of traceability models in DW development process

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 automatic ones, since they will act as a bridge to map certain requirements to the data sources.

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 sense, 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 the ones 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 succesive deriving, adding, and filtering elements while most elements are not altered, the traces have low volatility. In this sense, developing a reactive framework which automatically updates the corresponding traces whenever a change (update or delete) is made would minimize the maintenance effort.

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 [20] rules.

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. Due to paper constraints, we will focus on the traces coming from CIM to PIM, both in this section and in our example.

According to our proposal for developing DWs [15, 16], 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 fact and fact attribute elements, is shown in figure 5.

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

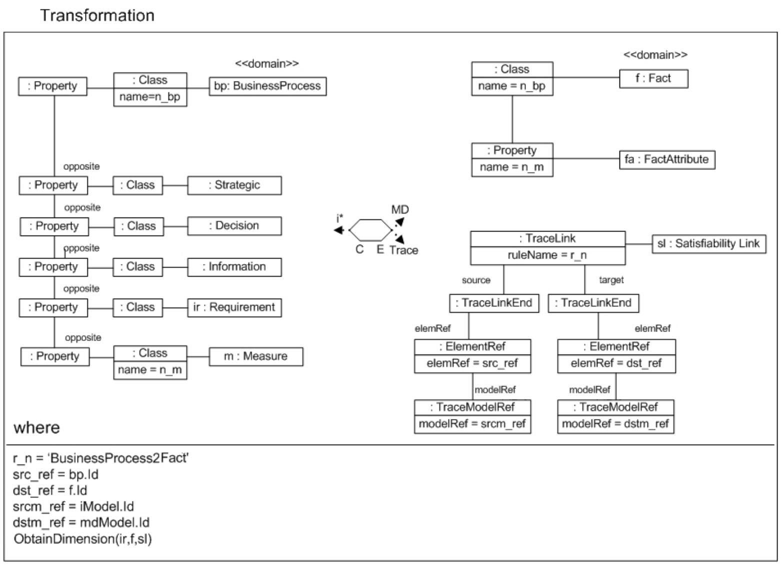

Fig. 5. QVT rule to derive a fact from a business process along with the associated trace

On the right hand of the transformation rule are the target metamodels. On the one hand, we have our multidimensional profile, composed by the fact (focus of the analysis, related to the business process) and the associated fact attribute (indicator of performance, related to the measure). On the other hand we also have the trace metamodel we proposed, composed in this case by the trace link which makes explicit the relationship between the business process and the fact. In an analogous way, the trace link for making explicit the relationship between the measure and the fact attribute would be also created, but due to space constraints it is omitted.

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. The rest of the QVT relationships between models in our approach (without including traceability) can be found in [16]. The corresponding QVT rule for transforming a context into a dimension can be seen in figure 6.

In this figure, a context “c” from the CIM, on the left hand, is transformed into a dimension “d” and its associated base level “b” in the PIM, at the right hand of the figure. The associated trace link to this transformation has a source element reference, which is the context from the CIM, and three (one omitted) 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

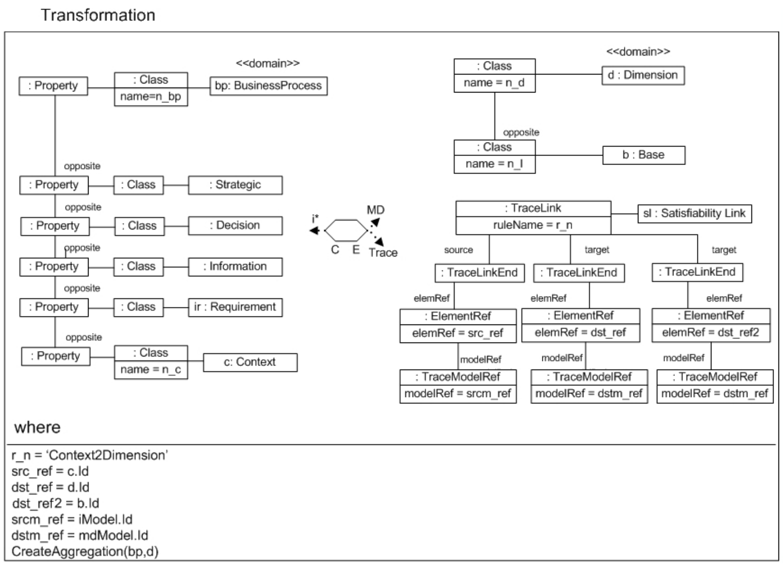

Fig. 6. QVT transformation for deriving a context into a dimension with its associated trace

omitted trace link corresponds to the association between the base level and the dimension. The rest of the relationships between the CIM and PIM elements is summarized in table 1, along with its corresponding rule name.

Table 1. Relationships between CIM elements and PIM elements

Transformation 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

The contexts can derive either into a dimension package, its corresponding dimension and first level if the context is the base of the dimension, or into a level and an association with the previous level in the dimension hierachy. A graphic example of the generic relationships is shown in figure 7.

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 succesive goals, which derive into an information requirement for

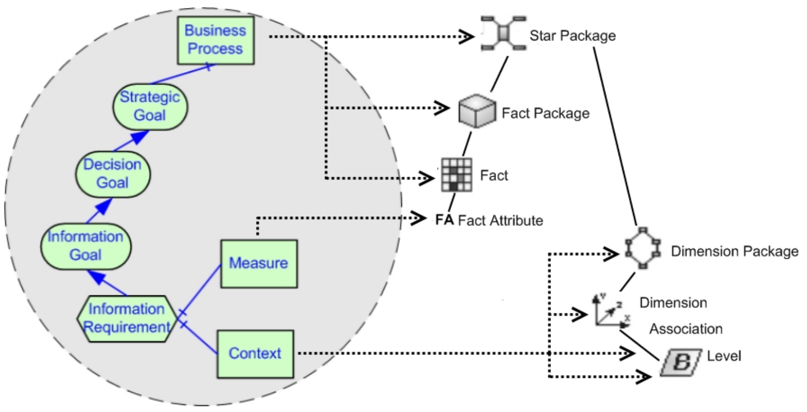

Fig. 7. Generic relationships between CIM elements and PIM Elements

the DW. In turn, this information requirement is decomposed into contexts and measures. At the right hand 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 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.

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.

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

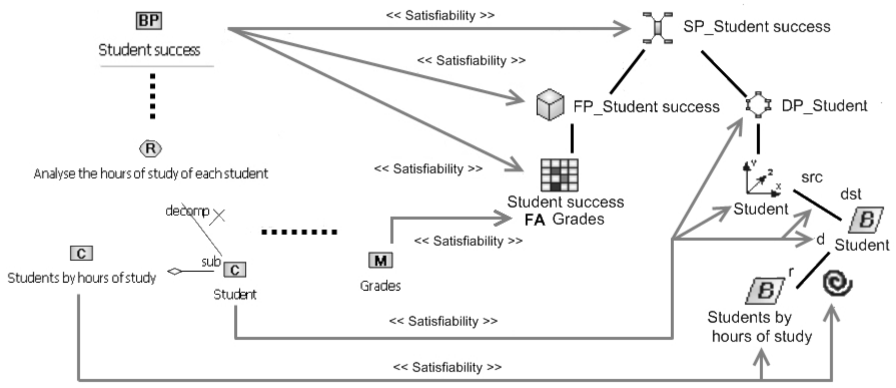

Fig. 8. Requirements modeled by using our i* profile for DW and its corresponding multidimensional model

per week. These requirements are recorded in a CIM which acts as the starting point for the process.

In figure 8, 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 hand 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.

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

Satisfiability link rule name	Source Elements (CIM)	Target Elements (PIM)
BusinessProcess2Fact	Student success	SP_Student sucess, FP_Student sucess, Student sucess
Measure2FactAttribute	Grades	Grades
Context2Dimension	Student	DP_Student,Student(Dime) Student(Level),Association
Context2Base	Students by hours of study	Students by hours of study (Level), Association

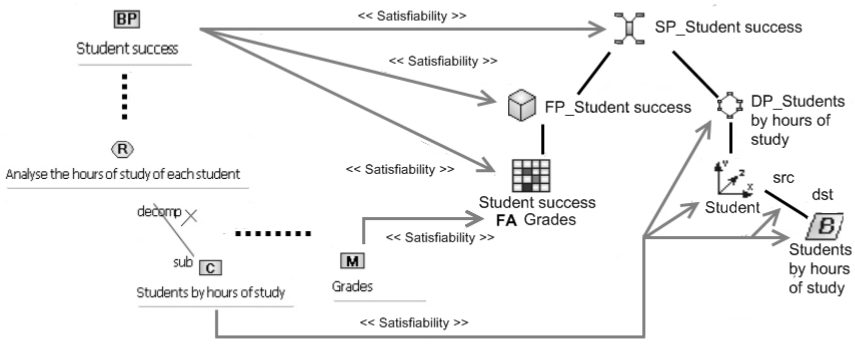

Fig. 9. New version of the CIM and its corresponding PIM obtanied by applying our trace metamodel and the transformations

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 figure 9.

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 forty to fifty decisional goals, deriving into an average of twenty eight to thirty four 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 Conclusions and Future Work

In this paper, we have proposed the first 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 was 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. We will also develop a traceability framework in order to make the maintenance of traces as automatic as possible.

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 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. Arkley, P., Mason, P., Riddle, S.: Position paper: Enabling traceability. In: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, Edinburgh, Scotland, pp. 61–65 (2002)
4. Ballou, D., Tayi, G.: Enhancing data quality in data warehouse environments. Communications of the ACM 42(1), 73–78 (1999)
5. Barbero, M., Del Fabro, M., Bézivin, J.: Traceability and provenance issues in global model management. In: ECMDA-TW, pp. 47–56 (2007)
6. Del Fabro, M., Bézivin, J., Valduriez, P.: Weaving Models with the Eclipse AMW plugin. In: Eclipse Modeling Symposium, Eclipse Summit Europe 2006, Esslingen, Germany (2006)
7. Franch, X.: Incorporating Modules into the *i** Framework. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 439–454. Springer, Heidelberg (2010)
8. Gotel, O., Finkelstein, A.: An analysis of the requirements traceability problem. In: ICRE, pp. 94–101. IEEE, Los Alamitos (1994)
9. 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)
10. Inmon, W.H.: Building the data warehouse. Wiley-India, Chichester (2009)
11. Jouault, F.: Loosely coupled traceability for atl. In: ECMDA-TW, Nuremberg, Germany, pp. 29–37 (2005)
12. Kimball, R.: The data warehouse toolkit. Wiley-India, Chichester (2009)
13. Kolp, M., Giorgini, P., Mylopoulos, J.: Organizational Patterns for Early Requirements Analysis. In: Advanced Information Systems Engineering (CAiSE), LNCS, vol. 2681, pp. 617–633. Springer Berlin (2003)

14. Luján-Mora, S., Trujillo, J., Song, I.-Y.: A UML profile for multidimensional modeling in data warehouses. DKE 59(3), 725–769 (2006)
15. Mazón, J.-N., 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)
16. Mazon, J.N., Pardillo, J., Trujillo, J.: An MDA approach for the development of data warehouses. DSS 45(1), 41–58 (2008)
17. Mazón, J.-N., Trujillo, J., Lechtenbörger, J.: Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. DKE 63(3), 725–751 (2007)
18. Mouratidis, H., Giorgini, P., Manson, G.: Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 63–78. Springer, Heidelberg (2003)
19. OMG: A Proposal for an MDA Foundation Model (2005)
20. OMG: The Meta-Object Facility 2.0 Query/View/Transformation. Final Adopted Specification (2005)
21. Giorgini, P., Rizzi, S., Garzetti, M.: GRAnD: A goal-oriented approach to requirement analysis in data warehouses. DSS 45(1), 4–21 (2008)
22. Paige, R., Olsen, G., Kolovos, D., Zschaler, S., Power, C.: Building model-driven engineering traceability classifications. In: ECMDA-TW, pp. 49–58 (2008)
23. Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Transactions on Software Engineering 27(1), 58–93 (2001)
24. Ramesh, B., Stubbs, C., Powers, T., Edwards, M.: Requirements traceability: Theory and practice. Annals of Software Engineering 3(1), 397–415 (1997)
25. Samia Kaabi, R., Souveyet, C., Rolland, C.: Eliciting service composition in a goal driven manner. In: ICSOC, pp. 308–315 (2004)
26. Spanoudakis, G., Zisman, A.: Software traceability: a roadmap. Handbook of Software Engineering and Knowledge Engineering (2005)
27. Vassiliadis, P.: Data Warehouse Modeling and Quality Issues. Ph.D. thesis, Athens (2000)
28. Walderhaug, S., Stav, E., Johansen, U., Olsen, G.K.: Traceability in Model-Driven Software Development. Designing Software-Intensive Systems: Methods and Principle, 133–159 (2008)
29. Winkler, S., von Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Software and Systems Modeling 9, 529–565 (2010)
30. Yu, E.S.K.: Modelling strategic relationships for process reengineering. Ph.D. thesis, Toronto, Ont., Canada (1995)
31. Yu, Y., Jurjens, J., Mylopoulos, J.: Traceability for the maintenance of secure software. In: ICSM 2008, pp. 297–306. IEEE, Los Alamitos (2008)
32. Yu, Y., Niu, N., Gonzalez-Baixauli, B., Candillon, W., Mylopoulos, J., Easterbrook, S., do Leite, J., Vanwormhoudt, G.: Tracing and validating goal aspects. In: RE 2007, pp. 53–56. IEEE, Los Alamitos (2007)