Towards the effective use of traceability in Model-Driven Engineering projects

Iván Santiago, Juan M. Vara, Valeria de Castro, and Esperanza Marcos

Kybele Research Group, Rey Juan Carlos University,
Avda. Tulipán S/N, 28933 Móstoles, Madrid, Spain
{ivan.santiago,juanmanuel.vara,valeria.decastro,esperanza.marcos}@urjc.es
http://www.kybele.es

Abstract. The key role of models in any Model-Driven Engineering (MDE) process provides a new landscape for dealing with traceability. In the context of MDE traces are merely links between the elements of the different models handled along the software development cycle. Traces can be consequently stored in models that can be processed by means of model-based techniques. To take advantage of this scenario, this paper introduces iTrace, a framework for the management and analysis of traceability information in MDE projects. We present the methodological proposal bundled in the framework as well as the tooling that supports it. Finally, a case study is used to introduce some of the functionalities offered by the framework.

Keywords: Model-Driven Engineering, Traceability, Trace Models

1 Introduction

Despite the acknowledged significance that traceability should deserve in any Software Engineering activity [1], it is frequently ignored due to the inherent complexity of maintaining links among software artifacts [2]. The advent of Model-Driven Engineering (MDE, [3]) can drastically change this landscape. In particular, the key role of models can positively influence the management of traceability information since the traces to maintain might be simply the links between the elements of the different models handled along the process. Furthermore, such traces can be collected in other models to process them using any model processing technique, such as model transformation, model matching or model merging [4].

To confirm this assumption, in previous works [5] we have conducted a systematic review to assess the state of the art on traceability management in the context of MDE. One of the main conclusions derived from such review was that the operations most commonly addressed by existing proposals are storage, visualization and CRUD (creation, retrieval, updating and deletion) of traces. By contrast, those less commonly addressed are the exchange and analysis of traceability information. In particular, the analysis of traceability information was addressed only by 32.35% of the proposals analyzed.

To deal with the lack of proposals focused on the analysis of traceability information, this works introduces `iTrace`, a methodological and technical framework for the management of traceability information in MDE scenarios. As will be shown later, `iTrace` automates the production of trace models and supports

different types of analysis, at different levels of abstraction, of the raw data provided by such traces.

The rest of this paper is structured as follows: Section 2 reviews related works in the area; Section 3 presents the process for the production an analysis of trace models; Section 4 illustrates the proposal by means of a case study; and finally, Section 5 summarizes the main conclusions derived from this work and provide some directions for further work.

2 Related works

There are several works dealing with traceability in the context of MDE. Most of them deal with storage, visualization, semi-automatic generation, and CRUD operations for traces but only a few of them are focused on the analysis of traceability information. Those dealing with analysis can be grouped into three big categories attending to their main focus: 1) impact analysis [6] [7] [8]; 2) generation of traceability reports [9] [10] and 3) identification and classification of traces [11].

Regarding the works focused on impact analysis, the works from Anquetil *et al.* [6] and Walderhaug *et al.* [8] present proposals to extract information from trace links repositories in order to identify all the artifacts potentially impacted by a change. The former accompanies the proposal with a supporting tool. Likewise, Olsen and Oldevik in [7] presented a prototype where linked artifacts are displayed when source model elements are selected.

Next, we consider two different tools focused on the generation of traceability reports: `TraceTool` and Safety Requirements Trace Tool(`SRTT`). The former was presented by Valderas and Pelechano in [9]. It generates HTML traceability reports describing how the elements of a navigational model have been derived from a requirements model. The latter was presented in [10] by Sanchez *et al.*and consumes trace models to generate HTML traceability reports in the context of model-driven development of tele-operated services robots. Both tools are focused on requirements traceability, i.e. they do not consider low-level traceability, such as those produced when moving from design models to working-code.

Finally, in [11] Paige *et al.* present the Traceability Elicitation and Analysis Process (`TEAP`). `TEAP` elicits and analyze traceability relationships in order to determine how they fit into an existing traceability classification. Classifying trace links can help to understand and manage them. Although this proposal is very useful for MDE practitioners, it can be difficult to understand for a business analyst or an end-user.

Regarding the current state of the art, `iTrace` provides a number of contributions: first, it automates the production of traces in MDE projects. To that end, it bundles different components to inject traceability capabilities into existing model transformations. Next, it considers all the models involved in the development process and consequently it does does not limit its scope to requirements traceability. In addition, it provides new editors for trace models edition that fit better with the relational nature of trace models. Besides, all the works reviewed, except the one from Walderhaug *et al.* [8], supports ad-hoc analysis. That is to say, the analyzing process proposed fits in the particular MDE projects considered by each proposal. By contrast, `iTrace` aims at providing a generic proposal that can be applied to any given MDE project. Finally, one of the most relevant features of `iTrace` regarding existing proposals is the fact

that it supports multidimensional analysis from trace links. This feature will be later introduced in this paper.

3 Supporting the production and analysis of trace models

This section introduces our proposal to support the analysis of traceability information by outlining the main steps of the process and presenting the technical components that comprise the `iTrace` framework[1].

The starting point of the analysis process supported by `iTrace` is an existing MDE project. Note that we aim at defining a "project-agnostic" proposal, i.e. a proposal that can be applied to any type of project. Thus, to illustrate the idea we consider as starting point a generic MDE project, i.e. one that includes models at different abstraction levels (including weaving models) plus a set of model transformations connecting them (both m2m and m2t). This way, left-hand side of Fig. 1 shows a simple project, composed of three models, namely `Ma`, `Mb` and `Mc` and two m2m transformations connecting them (`Ma2Mb` and `Mb2Mc`).

The rest of Fig. 1 provides a tabular overview of the `iTrace` framework: the **iTrace Component** row shows the technical components supporting each step of the process, while the **Process** row shows the artifacts (models, transformations, etc.) handled. Finally, the **Notation** row shows a brief legend to ease the understanding of the different graphic elements.

During the **Identification** stage, an existing project composed of models and transformations is selected as input. In the **Addition** phase, the transformations are enriched by means of Higher-Order Transformations (HOTs) [12] following the idea first-sketched by Jouault in [13]. The enrichment process bundled in `iTrace` is a little bit more complex than the one from [13], due to the increased complexity of `iTrace` metamodels.

Next, the enriched transformations are run during the **Execution** phase in order to generate the corresponding target model/s plus the trace models that connect their elements. During the **Union** phase, another transformation consolidates such trace models into a unique `iTrace` model.

In some sense, the trace models generated at this point are *normalized* models. However, we have found *denormalized* trace models to be more convenient to run the queries that perform the Data-Driven Analysis supported by `iTrace`. This idea is brought from the databases area, where normalized data models are used for day-to-day activities (mainly CRUD operations), while denormalized models are preferred for read-intensive operations, like those supported by data-warehouses [14]. In some sense, the Data-Oriented Analysis of traces can be seen as the analysis of the historical data collected during the development of the project. These data are persisted in the shape of trace models. Therefore, during the **Transformation** phase, a m2m transformation consumes the normalized trace models generated in the Union phase to produce a denormalized trace model (`iTrace` denormalized model).

[1] Full screen images for all the figures can be found in `http://www.kybele.etsii.urjc.es/er/`

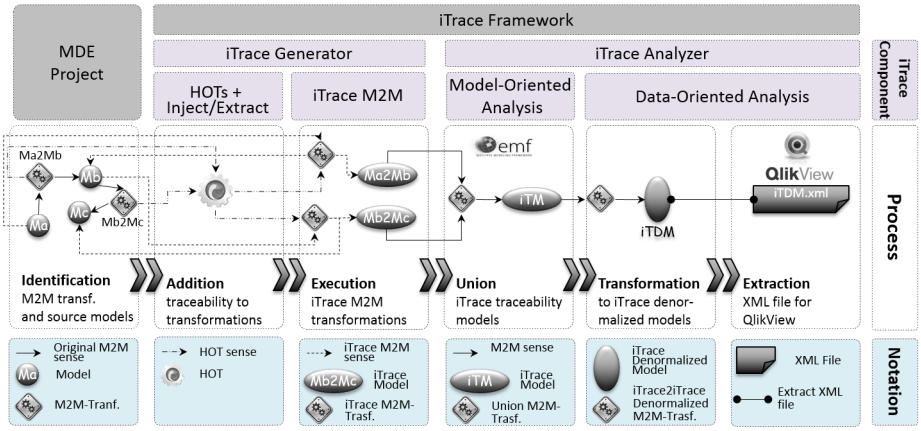


Fig. 1. iTrace process overview

Finally, during the **Extraction** phase, the iTrace denormalized model is serialized into an XML file that is later imported into QlikView[2] in order to populate the different dashboards used to support the Data-Oriented Analysis provided by `iTrace`Às a result, high-level overviews of the relationships between the artifacts involved in the development process are obtained from the trace models produced in the project. Next section illustrates the idea by showing two particular scenarios of Data-Oriented Analysis by means of a case study.

4 Case Study

As previously mentioned, iTrace aims at improving the use of traceability information in MDE projects. This section shows how it is effectively done for an existing MDE project. To that end, we lean on M2DAT-DB [15], a framework for Model-Driven development of modern database schemas that support the whole development cycle, from PIM to working code. In particular, `M2DAT-DB` supports the generation of ORDB schemas for Oracle and the SQL:2003 standard as well as XML Schemas from a conceptual data model represented with a UML class diagram. In this work we focus on three of the model transformations bundled in `M2DAT-DB`: the UML2SQL2003 transformation produces a standard ORDB model from a UML class diagram; the SQL20032ORDB transformation generates an ORDB model for Oracle from the standard one; finally, the ORDB2SQL2003 transformation implements the inverse transformation. Besides, apart from the corresponding source models, each transformation is able to consume an annotation model to introduce some design decisions in the transformation process.

In the project under study such transformations are run to produce the Online Movie Database (OMDB) schema. The OMDB is a case study presented by Feuerlicht *et al.* [16] that has been also used by some other authors. Using an "external" case study prevents us from using ad-hoc models that might fit better to our purposes. The OMDB is devised to manage information about movies, actors, directors, playwrights and film-related information.

[2] QlikTech International AB. `http://www.qlikview.com`

Next sections present two of the dashboards provided by `iTrace`. They are populated with the data collected in the `iTrace` models produced when `M2DAT-DB` is used to generate the OMDB schema. First dashboard can be used to get an overview of the transformations involved in the process. Second dashboard provides a deeper view of the mapping rules and their workload.

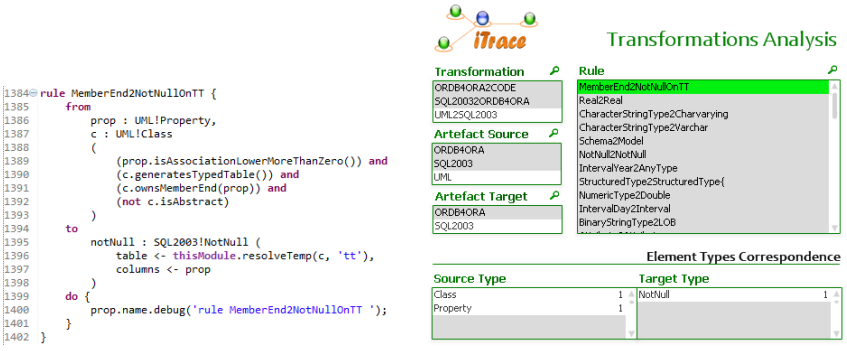
4.1 Mapping rules overview

Model transformations are inherently complex [17]. Therefore, dealing with legacy transformations might be even more complex. The analysis of trace models can be used in this context to raise the abstraction level at which they think about model transformations. In addition, it allows the developer to abstract from the particular model transformation language used and provide him with simple and comprehensible information regarding the mapping rules that compose the different transformations.

For instance, Fig. 2-a shows a code excerpt from the `UML2SQL2003` transformation. In particular, it shows the `MemberEnd2NotNullOnTT` mapping rule. To understand the functionality of this rule and what type of elements it maps, a minimum knowledge of ATL is needed.

By contrast, a quick look at the dashboard beside shows at first sight which the purpose of the mapping rule is. Indeed, no previous knowledge of ATL is needed. The information displayed on the upper side of the dashboard reveals that the rule maps objects from UML models into SQL2003 objects. In particular, the lower side of the dashboard shows that the rule is responsible for mapping pairs of `Class` and `Property` objects into `NotNull` objects.

Note that the dashboard provides this type of information for all the transformations that conform the project under study. The upper side of the dashboard provides a set of filters that allow the user to select a particular transformation, source model, target model or mapping rule/s (non-selected filtering values are greyed-out). The bottom part of the dashboard shows the type of elements transformed by the mapping rules that meet the criteria established by the user. Note also that this analysis may be useful in order to document not only m2m transformations, but also m2t ones.



(a) Source code (b) Dashboard-supported overview

Fig. 2. Two ways of looking at the `MemberEnd2NotNullOnTT` mapping rule

4.2 Mapping rules workload

Previous section has shown how trace models can be processed to provide technology-independent overviews of the model transformations involved in a given project. The dashboard shown in Fig. 3 goes a step further, since it provides a closer look at the transformations. In particular, it allows identifying which are the rules involved in a given transformation and which is their role in the project. The latter refers to the workload of such rules, i.e. the amount of objects effectively mapped by the mapping rule under consideration.

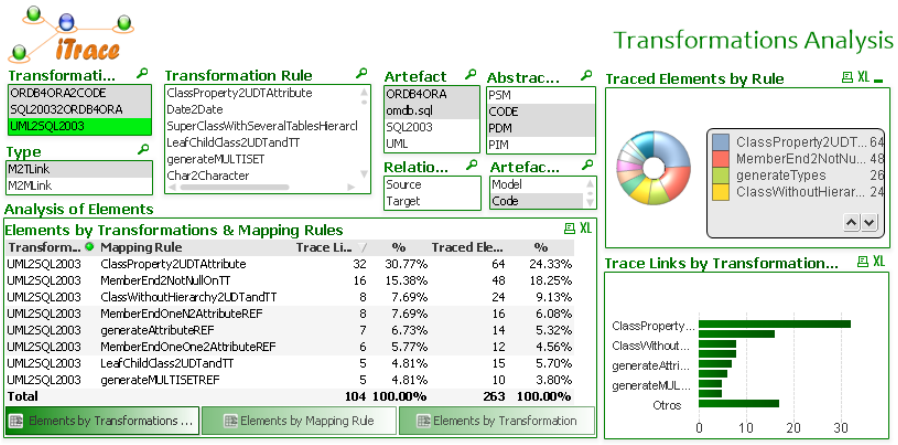


Fig. 3. Mapping rules workload dashboard

To that end, upper side of the dashboard bundles a set of controls to define high-level criteria for the analysis. This way, the traces that will be analyzed can be filtered according to:

- Transformations: only traces produced by the select model transformations will be considered.
- Type: only model-to-model or model-to-text traces will be considered.
- Transformation Rule: only traces produced by the selected mapping rules will be considered.

Likewise, the model elements that will be object of consideration can be filtered according to another set of criteria:

- Artefact: only elements included in the selected models or source-code files will be considered.
- Relation Type: depending on the selection, only model elements used that were either as source or target objects of the selected transformations will be considered.
- Abstraction Level: only model elements belonging to models defined at the selected abstraction levels will be considered.
- Artefact Type: depending on the selection, only model elements or source-code blocks will be considered.

Obviously, none of the criteria above are mandatory. That is to say, the user might set no values for any of them. If so, no filtering is applied and every trace (respectively model element) is considered in the analysis.

Once the criteria have been fixed (if any), the central and lower part of the dashboard collects aggregated data regarding the number of traces, model elements (referred to as `traced elements`) and source-code blocks, which fulfill the criteria. In this case, the table in the middle shows which are the mapping rules producing more traces. In particular, it shows the top 8 rules, while the rest are blended into the `Others` row.

First and the second columns show respectively the transformations and mapping rules rules under consideration (those that meet the filtering criteria). Next columns show the number of trace links produced and model elements referenced by each mapping rule and the percentage over the total number of traces produced by the mapping rules selected. For instance, second row of the table states that the `MemberEnd2NotNullOnTT` of the `UML2SQL2003` transformation generates 16 trace links (15.38% over the total number of trace links produced by the selected mapping rules) and such links refer to 48 model elements (note that not every trace link represents a one-to-one relationship).

Finally, the right side of the dashboard provides different views of these data. The pie chart represents the distributions of traced elements by transformation rule, while the bar graph summarizes the number of trace links produced by each transformation rule. As previously suggested, the information collected in this dashboard could be used by model-transformation developers to locate candidates for refining. For instance, the data presented in Fig. 3 highlights the importance of the **ClassProperty2UDTAttribute** mapping rule, which generates more than 30% of the trace links produced by the `UML2SQL2003` transformation. Thus, this rule might be object of study if transformation developers aims at optimizing the overall performance of the transformation.

5 Conclusion

This paper has introduced `iTrace`, a model-driven framework for the management of traceability information in MDE projects. The framework supports the production of trace models from any existing project and the subsequent analysis of such models. For instance, the dashboards from the case study have shown that the information provided by `iTrace` can be used to produce a high-level overview of the transformations involved in a project, to explain the purpose of a particular mapping rule or to identify the rules that should be optimized in order to improve the execution of a given transformation.

Two main lines are distinguished regarding directions for further work. On the one hand, we are extending the proposal to support other transformation languages in order to show that it can be effectively used with any metamodel-based language. Besides, we are integrating support for text-to-model transformations so that Model-Driven Reverse Engineering [3] projects can be also object of study.

Acknowledgments. This research is partially funded by the MASAI project, financed by the Spanish Ministry of Science and Technology (Ref. TIN2011-22617).

References

1. Asunción, H.U.: Towards practical software traceability. In: Companion of the 30th international conference on Software engineering. ICSE Companion '08, New York, NY, USA, ACM (2008) 1023–1026
2. Oliveto, R.: Traceability Management meets Information Retrieval Methods - Strengths and Limitations. In: 12th European Conference on Software Maintenance and Reengineering, CSMR. (2008) 302–305
3. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer **39** (2006) 25–31
4. Bernstein, P.: Applying model management to classical meta data problems. In: First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA (2003) 1–10
5. Santiago, I., Jiménez, A., Vara, J.M., De Castro, V., Bollati, V., Marcos, E.: Model-Driven Engineering As a New Landscape For Traceability Management: A Systematic Review. Information and Software Technology **54** (2012) 1340–1356
6. Anquetil, N., Kulesza, U., Mitschke, R., Moreira, A., Royer, J., Rummler, A., Sousa, A.: A model-driven traceability framework for software product lines. Software and Systems Modeling **9** (2010) 427–451
7. Olsen, G., Oldevik, J.: Scenarios of traceability in model to text transformations. In: Proceedings of the 3rd European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'07), Haifa, Israel (2007) 144–156
8. Walderhaug, S., Johansen, U., Stav, E., Aagedal, J.: Towards a generic solution for traceability in mdd. In: European Conference on Model-Driven Architecture - Traceability Workshop (ECMDA-TW'06), Bilbao, Spain (2006) 41–50
9. Valderas, P., Pelechano, V.: Introducing requirements traceability support in model-driven development of web applications. Information and Software Technology **51** (2009) 749–768
10. Sánchez, P., Alonso, D., Rosique, F., Álvarez, B., Pastor, A., Pastor, J.A.: Introducing Safety Requirements Traceability Support in Model-Driven Development of Robotic Applications. IEEE Transactions on Computers **60** (2011) 1059–1071
11. Paige, R., Olsen, G., Kolovos, D., Zschaler, S., Power, C.: Building model-driven engineering traceability classifications. In: Proceedings of the 4th European Conference on Model Driven Architecture - Traceability Workshop (ECMDA-TW'08), Berlin, Germany (2008) 49–58
12. Tisi, M., Cabot, J., Jouault, F.: Improving higher-order transformations support in ATL. In: International Conference on Model Transformation. (2010) 1–10
13. Jouault, F.: Loosely coupled traceability for atl. In: Proceedings of the European Conference on Model Driven Architecture (ECMDA) Workshop on Traceability, Nuremberg, Germany. Volume 91. (2005)
14. Sanders, G.L., Shin, S.: Denormalization Effects on Performance of RDBMS. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Vol.3, Washington, DC, USA, IEEE Computer Society (2001)
15. Vara, J.M., Marcos, E.: A framework for model-driven development of information systems: Technical decisions and lessons learned. Journal of Systems and Software **85** (2012) 2368 – 2384
16. Feuerlicht, G., Pokorny, J., Richta, K.: Object-Relational Database Design: Can Your Application Benefit from SQL:2003? In: Information Systems Development. Springer (2009) 975–987
17. Bollati, V.A., Vara, J.M., Jiménez, Á., Marcos, E.: Applying MDE to the (semi-)automatic development of model transformations. Information and Software Technology **55** (2013) 699 – 718