Towards a Generic Traceability Framework for Model-driven Software Engineering

Birgit Grammel

SAP Research CEC Dresden, Chemnitzer Str. 48
01187 Dresden, Germany

Abstract. With the inception of Model-Driven Software Engineering (MDSD) the need for traceability is raised to understand the complexity of model transformations and overall to improve the quality of MDSD. Using the advantage of generating traceability information automatically in MDSD, eases the problem of creating and maintaining trace links, which is a labor intensive task, when done manually. Yet, there is still a wide range of open challenges in existing traceability solutions and a need to consolidate traceability domain knowledge. This paper proposes a generic framework for augmenting arbitrary model transformation approaches with a traceability mechanism. Essentially, this augmentation is based on a domain-specific language for traceability providing the formalization on integration conditions needed for implementing traceability. The paper is of positional nature and outlines work currently in progress.

1 Introduction

In the Standard Glossary of Software Engineering Terminology[1] traceability is defined as: *The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another.* Traceability information in MDSD can be understood as the runtime footprint of model transformations. Essentially, trace links provide this kind of information by associating source and target model elements w.r.t. the execution of a certain model transformation. The corresponding models may conform to the same or different metamodels. Trace links have a manifold application domain [1], [2]: *System analysis* to understand system complexity by navigating via trace links along model transformation chains; *Coverage analysis* to determine whether all requirements were covered by test cases in the development life cycle; *Change impact analysis* to analyze how changing one model would effect other related models; *Orphan analysis* to find orphaned model elements with respect to a specified trace link; *Model-based debugging*, i.e. mapping the stepwise execution of an implementation back to its high-level model and *Model transformation debugging*.

In this paper, we propose a generic traceability framework for augmenting arbitrary model transformation approaches with a traceability mechanism. To achieve this goal, we advocate two research directions: a) the augmentation of the generator logic, for the

[1] Std 610.12-1990

Grammel B. (2009).
Towards a Generic Traceability Framework for Model-driven Software Engineering.
In *Proceedings of the 1st International Workshop on Future Trends of Model-Driven Development*, pages 44-47
DOI: 10.5220/0002185400440047

sake of trace link generation b) the augmentation of input models with model markers, serving as a reference for trace link generation. Both approaches are based on a domain-specific language for traceability, presenting the formalization on all adequate conditions and measures needed for implementing traceability in the context of MDSD.

The content of the following sections are structured as follows. First, we reflect on challenges for traceability and give an overview on related work. In section 3 an outline of our approach is proposed. Finally, section 4 concludes this paper and gives an outlook on future work.

2 Challenges for Traceability in MDSD

MDSD raises the need for traceability to understand model transformation complexity and overall to improve the quality of MDSD. Introducing model transformations, to generate trace links automatically, eases the task of creating and maintaining trace links, as opposed to when done manually, entailing high development costs. [3]

According to [2], [4] transformation approaches either generate trace links implicitly or explicitly, that is, in the former case, either provide an integrated support for traceability or in the latter one, rely on a developer to encode traceability as a regular output model. The main advantage of implicit generation (e.g. QVT[2], MOFScript[3]) is, that no additional effort is necessary to generate trace links, as this is done automatically in parallel to the actual transformation. A disadvantage is, that the traceability metamodel is fixed. Since most transformation approaches have differently defined metamodels, standardization among different approaches is aggravated. Furthermore, the level of granularity of trace links may differ form one traceability scenario to another. Setting the granularity of trace links has been identified as a challenge [5]. When tracing all model element references, this might lead to the following issues: *incomprehensibility* of trace link data and hence be less useful to developers; *performance*, when handling large and complex transformations and *security*, when not all model information is allowed to be traced for security reasons, mandated for instance by customer needs.

Alternatively, the case of explicit generation necessitates the incorporation of additional transformation rules to generate trace links. (e.g. ATL[4], oAW[5]) The metamodel definition is not predefined and transformation engine independent. Hence, the trace link granularity is adaptable. Yet, the drawback is that additional effort is required to add traceability-specific transformation rules, which may also pollute the implementation. An approach that partly solves these issues in ATL by automatically generating traceability-specific transformation rules was proposed in [6].

Another challenge concerns the semantics of trace links [5]. It is often necessary to distinguish between different kinds of links, e.g. a link between a textual requirement and a model element has a different semantic, than a refinement relationship within a model. Fixing the kinds of semantic links, has the consequence of less flexibility for

[2] Query View Transformation, http://www.omg.org/docs/ptc/07-07-07.pdf
[3] http://www.modelbased.net/mofscript/
[4] ATLAS Transformation Language, http://www.eclipse.org/m2m/atl/
[5] openArchitectureWare, http://www.openarchitectureware.org/

user-defined links that might be necessary to meet different project needs [7]. On the other hand, since the choice of semantic, attributed to a link, is guided by the reasoning about what the user will perform with the link [1], not predefining the link semantics, may result in failure of reasoning, due to misuse of the semantics.

3 Traceability Approach

The main idea of our approach is to rely on work and solutions already available and not to implement yet another transformation language. Nevertheless, is the aim to focus on current traceability solutions, consolidate benefits of implicit and explicit trace link generation and tackle the above-mentioned challenges to derive the necessary requirements for a generic traceability framework. In essence, our approach follows two research directions both based on a commonly used conceptional layer. The main motivation for such a conceptional layer is based on the idea, proposed in [7], that the kind of traceability data to be collected is dependent on the actual traceability goal, that is, which traceability question is expected to be answered. To formalize this traceability concern, a system of rules and regulations is necessary, on when to trace, what kind of traceability data; additionally, to specify, the granularity level and what kind of trace links are allowed between certain artefact types, given a certain traceability goal. This formalization will be defined w.r.t a domain specific language for traceability, which is called Trace-DSL in the following. The Trace-DSL logic encompasses all heuristics on how to manifest traceability-specific rules for arbitrary model transformation approaches, providing explicit knowledge to the developer on their integration.

To create a traceability model we discussed two classes of transformation approaches in section 2. Explicit trace link generation requires the incorporation of additional transformation rules to generate trace links. As this task is generally done manually, it is likely to be error-prone and time consuming. Although work has been done on the automation, e.g. [6], still every transformation template needs to be adapted individually, for the explicit generation. For generators, that already provide a dedicated traceability support, e.g. QVT, this effort would theoretically not be necessary, yet there is a need to tune the level of granularity of trace links, as motivated in section 2. Native tracing without a clear rule system on the conditions and measures of tracing, may lead to an unnecessary overkill of traceability information, performance issues and violation of security measures. The guidance, to trace only relevant information is provided by the Trace-DSL.

Hence, the following two research directions are investigated, showing different approaches on how to achieve the population of a traceability metamodel. The first research direction (Fig.1a) is proposed to enhance the generator logic itself for the purpose of traceability functionality. a) **Augmentation of the generator logic**: One way to populate a traceability metamodel is to provide a generic interface and trace engine for arbitrary generators. In this case the interface supplies the engineer with an API to connect his generator to the trace engine. As a result, the generator is featured with traceability functionality. The interface is based on the Trace-DSL, to account for a formalized rule system on traceability. The second approach (Fig.1b) to be examined follows the idea of high order transformations and in contrast to the first approach does not focus on the

level of the generator logic. b) **Augmentation of input and output models**: Prior to the actual transformation the input model is augmented via model transformations with certain markers such as identifiers or properties. This enhanced model is then transformed by the generator into an enhanced output model. In essence, both models are compared automatically w.r.t. the correlation of before and after markers. Querying of traceability information is then coupled to these methods of auto-correlation. The Trace-DSL orchestrates the model augmentation to again only trace relevant information for a certain underlying traceability question.

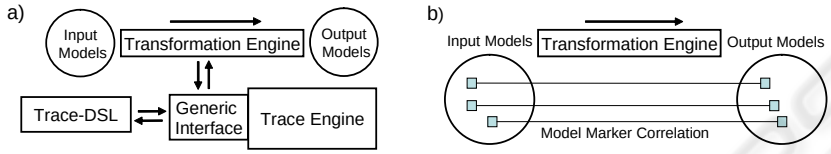

Fig. 1. Traceability research approaches.

4 Conclusions and Future Work

The two research directions run orthogonally to each other w.r.t. their target points, being the generator logic level versus the model level. By comparing their complexities, advantages and disadvantages, we wish to develop a generic traceability framework for arbitrary transformation approaches.

Future work is directed at the generation of traceability-specific rules based on the principles of metamodel matching [8]. It is to be investigated, how to use the resulting alignment of metamodel matching, to generate the code of traceability-specific rules, automatically.

References

1. Ramesh, B., Jarke, M.: Towards reference models for requirements traceability. In: IEEE Transactions on Software Engineering. Volume 21. (2001)
2. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal, Vol 45, No 3 (2006)
3. Aizenbud-Reshef, N., et al.: Model traceability. IBM Systems Journal, Vol 45, No 3 (2006)
4. Vanhooff, B., Baelen, S.V., Joosen, W., Berbers, Y.: Traceability as input for model transformations. In: Proceedings of the ECMDA Traceability Workshop. (2007)
5. Antoniol, G., et al.: Problem statement and grand challenges in traceability. Center of Excellence for Traceability (2006)
6. Jouault, F., Kurtev, I.: Transforming models with ATL. In: Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005. (2005)
7. Knethen, A., Paech, B.: A survey on tracing approaches in practice and research. Technical report, Frauenhofer IESE (2002)
8. Falleri, J., Huchard, M., Lafourcade, M., Nebut, C.: Metamodel matching for automatic model transformation generation. In: Proceedings of MoDELS. (2008)