UNIVERSITY OF GOTHENBURG

Evaluating Model Transformation Technologies

An exploratory case study

Bachelor of Science Thesis in Software Engineering and Management

K. M. ARIF AZIZ

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, May 2011

Evaluating Model Transformation Technologies

An exploratory case study

K. M. ARIF AZIZ

Examiner: HELENA HOLMSTRÖM OLSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, May 2011

Evaluating Model Transformation Technologies

K. M. Arif Aziz
Software Engineering and Management
University of Gothenburg
Department of Computer Science and Engineering
Gothenburg, Sweden
arifaziz@student.gu.se

Abstract

Model transformation is one of the primary activities within model driven software engineering. Several model transformation languages have been proposed and implemented providing different approaches, programming paradigms, and tools to solve common tasks related to model transformations. While several of these languages exist, little guidance is provided to software industries to select a language that will suit their needs. This paper presents the results of a case study of exploring three model transformation technologies at a department of Ericsson AB. The findings of this study can be used when judging the applicability of a particular model transformation technology.

1 Introduction

The application of Model Driven Software Engineering (MDE) is growing within software industry. The major vision of MDE is that *models* are used as primary artefacts during different phases of software development starting from requirements elicitation through implementation and deployment to maintenance. MDE enables development using concepts closer to the problem domain, leads further towards automation, and allows work at a higher level of abstraction.

Model driven architecture (MDA) is one approach to MDE. The primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns [Miller, 2003; Kleppe et al., 2003]. MDA supports specification of the operation of a system being separated from the details of its platform. One first specifies the system independently of the platform; the specification is then *transformed* for the given platform. Specifications of systems and platforms are expressed in *modeling languages* (e.g. UML, EMF, SysML), and transformations in *transformation languages*. Several transformation languages have been proposed and implemented (e.g. ATL, QVT, Java APIs) providing different approaches, programming paradigms, and sets of tools to solve common tasks related to *model transformations*.

However, little guidance has been provided for software industries adopting model transformation. Model transformation technologies are often sophisticated, vary in ease of use, capacity, and require expensive training. Software industries need support for choosing suitable model transformation technologies for their MDA projects.

This study looks into qualities of model transformation technologies from an industrial perspective. The paper aims to answer the following research questions:

- RQ1: What qualities of available model transformation technologies are important to MDA engineers in the software industry?
- RQ2: To which degree do available model transformation tools have these qualities?
- RQ3: Which of the found qualities (in RQ1) are most important to MDA users in industry?

The research uses the *exploratory case study methodology* [Benbasat et al., 1987; Easterbrook et al., 2008; Runeson and Höst, 2009; Yin, 2003]. The study was performed at a department of Ericsson AB, Gothenburg, Sweden, which uses model driven development as its core for software development. Ericsson is a leading provider of telecommunication and data-communication systems for the international market. This paper presents the findings of the case study in the form of *a quality model, a quality rank, and a transformation technology ranking* for the transformation technologies used in the study.

The paper is organized as follows: Section 2 describes the design of the case study. The main principles of MDA are discussed in Section 3, while the results of the study are presented in Section 4. Section 5 provides with discussion. Section 6 presents the related work. In Section 7 conclusions are drawn and future work is suggested.

2 Research Design

We used *exploratory case study* as a method for this study [Benbasat et al., 1987; Easterbrook et al., 2008; Runeson and Höst, 2009; Yin, 2003]. The research was a single case study with a real world model transformation problem as *the case* and model transformation technologies as its *embedded units of analysis* [Yin, 2003]. In the study, we investigated a certain phenomena (qualities important to the industry), built a theory (a quality model for model transformation technologies) while collecting qualitative data (interviews) to validate the theories and our findings.

2.1 Research Site

The research was performed at a department of Ericsson AB, Gothenburg, Sweden. The department required a model transformation solution to automate the transformation of their models (expressed in the Unified Modeling Language (UML) and UML profiles) to target code in a variety of programming languages. Before this study, a project was performed that provided a solution to the model transformation problem: a program written in a general purpose programming language (Java). The department grew interested in other model transformation solutions that could be applied, but faced the need to evaluate the qualities of the model transformation technologies first.

2.2 Research Process

As suggested by Runeson and Höst [2009], the research design for an exploratory case study should be flexible. We adopted the idea of Runeson and Höst [2009] and kept the research design open within the study period. However, this said, we had several distinct activities in the study (See Figure 1). The first activity was *Identification*, followed by *Exploration* where we had exploratory iterations over the embedded units of analysis (i.e., one iteration per technology). Research then moved on to *Evaluation* activity where we built a quality model through reflections on our experiences from Exploration. The final phase of the study was *Validation* through feedback (presentation and interviews) from the organization.

2.2.1 Identification

We identified available model transformation technologies and selected ones that were of interest to the department and that appeared to have the most potential (See Section 4.1 for the choice of technologies). We studied works within the field of transformation engineering. We gathered necessary artefacts, existing model transformation solution (written in Java), existing documentation of the transformation, exemplar output of from the existing transformation, and models in UML and UML profiles used as input for transformations. These UML models were actual solutions within the domain of telecommunication (real world model transformation problem).

2.2.2 Exploration

We selected one tool from the list of model transformation technologies for an iteration. We gathered and studied the documentation of the selected technology. Next, we tried the examples used in the documentation to be familiar with the approach, the transformation language, the user interface, and the workflows of the technology.
Once we learned the technology, we started implementing a model transformation solution for the case using the MDA approach. For the purpose of later theory building, we logged our experiences to act as a data source.

Termination Criteria: Within the Exploration activity, we needed to decide when a technology was sufficiently explored.
For model to model transformation technologies, we decided that the technologies should produce identical outputs (sufficiently explored). The time required to produce these identical outputs could be used as a data source for theory building.
For model to text transformation, we decided that the technologies should produce identical C++ source code of the existing model transformation solution.

2.2.3 Evaluation

Evaluation was our theory building activity. With ISO 9126 Quality Standards [ISO, 2001] as a starting point and through reflection of our experiences from the Exploration activity (Section 2.2.2), we constructed a quality model that recognized qualities seemingly important in the context of the case. As well, we made a quality scale and a model transformation technology ranking for the found qualities.

2.2.4 Validation

Once the quality model, and the quality scale were established, the model transformation technologies and the implemented transformations were presented to the engineers working with MDA at the department. The sessions were two hours each and consisted of the following: a demo in each technology for the studied case, a presentation of the implemented transformation, and a discussion of the

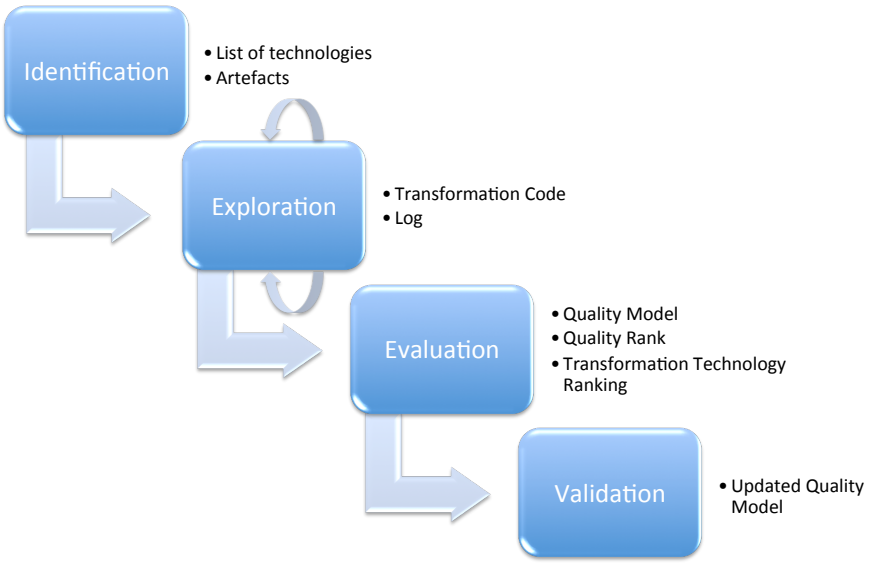


Figure 1: The figure summarizes the research process; the bullet list gives an overview of the outcomes for each activity.

quality model. The questions that were used to collect feedback were as follows:

1. Do you miss any qualities?
2. Which of these qualities in the quality model are of most importance to Ericsson AB? Could you rank them?
3. Could you also rank the technologies ?
4. Why are they ranked this way?
5. Which technology from model to model transformation or model to text (IBM, ATL, Java API, Acceleo) has the greatest potential from your perspective?
6. Is there anything that you would like to reflect more on Model Transformation?

The feedback collected was taken into account and the quality model, and the technology ranking was updated accordingly.

2.3 Threats to Validity

Any qualitative research approach will have threats to validity which may be internal or external to the research [Creswell, 2009; Yin, 2003]. During this study, we identified that our bias was potential threats to internal validity. Our preference to one technology over another could affect theory building. We handled threats to internal validity by not injecting our opinion during presentations.

3 Concepts

Model transformation involves well known concepts from the fields of MDE and MDA (See. e.g. Pilone and Pitman [2005]). Concepts include *meta-models* and *profiles*, *platform independent model (PIM)*, *platform specific model (PSM)*, *model transformation*, *model to model transformation (M2M)*, and *model to text transformation (M2T)*. (Note: We use summarized concepts of MDA in this study. Further concepts such as *marking models*, *PIM-PSM marks*, *PSM-code marks* [Kleppe et al., 2003] was not applicable for the particular case studied.)

3.1 Meta-model and Profiles

A *meta-model* typically defines the language from which to form a model. It provides syntax and semantics to express a model [Kleppe et al., 2003; Mens et al., 2005]. *Profiles* are the standard mechanism to extend meta-models (in UML). A profile is defined by a set of stereotypes, a set of related constraints, and a set of tagged values [Kleppe et al., 2003; Mens et al., 2005].

3.2 PIM

A *platform independent model (PIM)* is a model without information about implementation platform. This model is the result from *analysis activity* of software development using MDA. PIM is modeled at

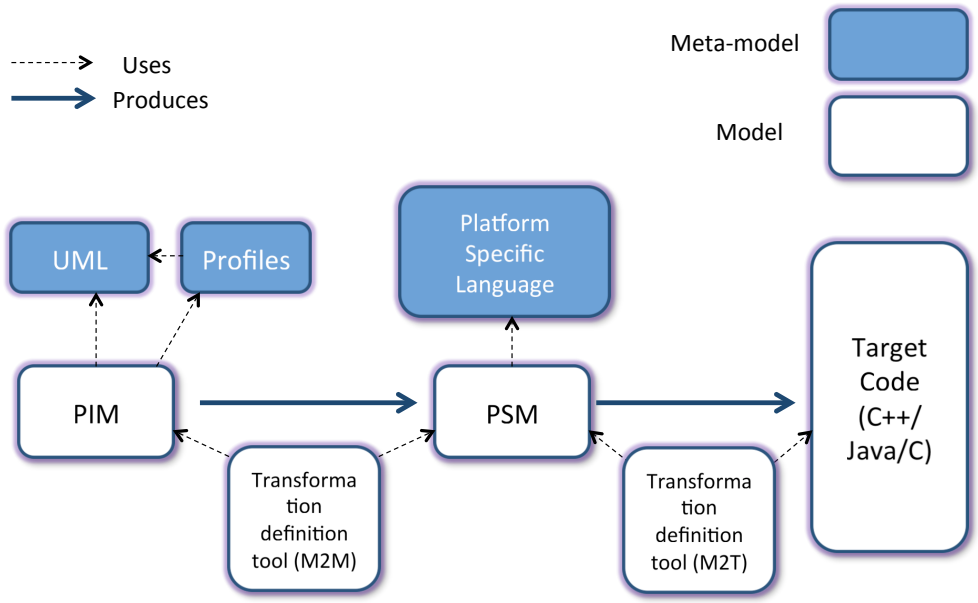


Figure 2: The figure presents the MDA approach summarized from Kleppe et al. [2003] (e.g. *PIM-PSM marks* [Kleppe et al., 2003] are not presented as this was not part of the study.)

a high level of abstraction that captures domain requirements. PIMs are usually expressed using UML and UML Profiles [Kleppe et al., 2003].

3.3 PSM

A *platform specific model (PSM)* injects the PIM onto a specific platform. PSM specifies a system in terms of the *software platform architecture*, and the implementation constructs of one technology (e.g. Java, C++, SQL, HTML). Many PSMs can be expressed for a single PIM. A *platform specific language* is typically engineered during *low-level design activity* of software development using MDA [Kleppe et al., 2003; Pilone and Pitman, 2005].

3.4 Model transformation

Model transformation is the automatic generation of a target artefact (PSM, code or documentation) from a source model (PIM), according to a transformation definition. A *transformation definition* is a set of transformation rules that together describe how a model in the source language is transformed into a model in the target language. *Transformation rules* describe of how one or more constructs in the source language map to one or more constructs in the target language [Kleppe et al., 2003].
MDA provides an architecture that guides the transformation of a PIM to its PSMs and the PSMs to target code or documentation for the system. This architecture is summarized within Figure 2.

3.4.1 M2M Transformation

In MDA, a *model to model transformation (M2M)* is the application of transformation definitions (set of transformation rules) to a PIM to obtain a PSM (expressed in the platform specific language). *M2M transformation tools* support writing transformation definitions, running transformations, and produce PSMs as outputs [Kleppe et al., 2003; Pilone and Pitman, 2005].

3.4.2 M2T transformation

In MDA, a *model to text transformation (M2T)* is a transformation definition (set of transformation rules) that transforms an expressed PSM to target source code or documentation (e.g. C++, Java, C#, Erlang, HTML). Similar to M2M tool, a *M2T transformation tool* allows writing transformation definitions, running transformations, and produce texts (target source code or documentation of a system) as outputs [Kleppe et al., 2003; Pilone and Pitman, 2005].

4 Results

4.1 Results from Identification

We selected the following technologies during Identification activity:

- *Atlas Transformation Language (ATL)* [LINA, 2011]. A report by Zeligsoft [Selic et al., 2010] which evaluated open source MDA technologies based on business criteria suggested ATL as a MDA tool having the highest potential within M2M technologies.
- *Acceleo* [Obeo, 2011]. The report by Zeligsoft [Selic et al., 2010] also suggested Acceleo as a MDA tool having the highest potential within M2T technologies.
- *IBM Transformation Framework (IBM TF)* [IBM, 2011]. The framework was part of the IBM Rational Software Architecture software suite used by the department at Ericsson AB.
- *Java Application Programming Interfaces (Java APIs)* [Eclipse, 2011a,b]. We collected data from a project at the department that used the Java APIs for M2M and M2T transformations.

4.1.1 ATL

ATL is a textual language for M2M transformations. ATL transformations are unidirectional, operate only on read-only source models and produce write-only target models [Jouault and Kurtev, 2006]. ATL is a hybrid language; it is a mix of declarative and imperative approaches to model transformation. It uses declarative constructs (rewrite rules, sets and theoretical computations) with imperative statements.

4.1.2 IBM TF

IBM TF is part of the IBM Rational Software Architecture software suite; it provides combinations of a visual editor (dominant) and a textual language (generated source code in Java) for M2M transformation. The visual editor allows to connect source model elements with target model elements to specify a transformation rule. Once a transformation is specified visually, Java source code is generated that implements the M2M transformations. The generated source code uses a collection of Java APIs to perform M2M transformation, such as UML Java API, and OCL Java API.

4.1.3 Acceleo

Acceleo is a text based M2T transformation technology. Transformation rules are written in templates that produce text files (source code or documentation) after execution.

4.1.4 Java APIs

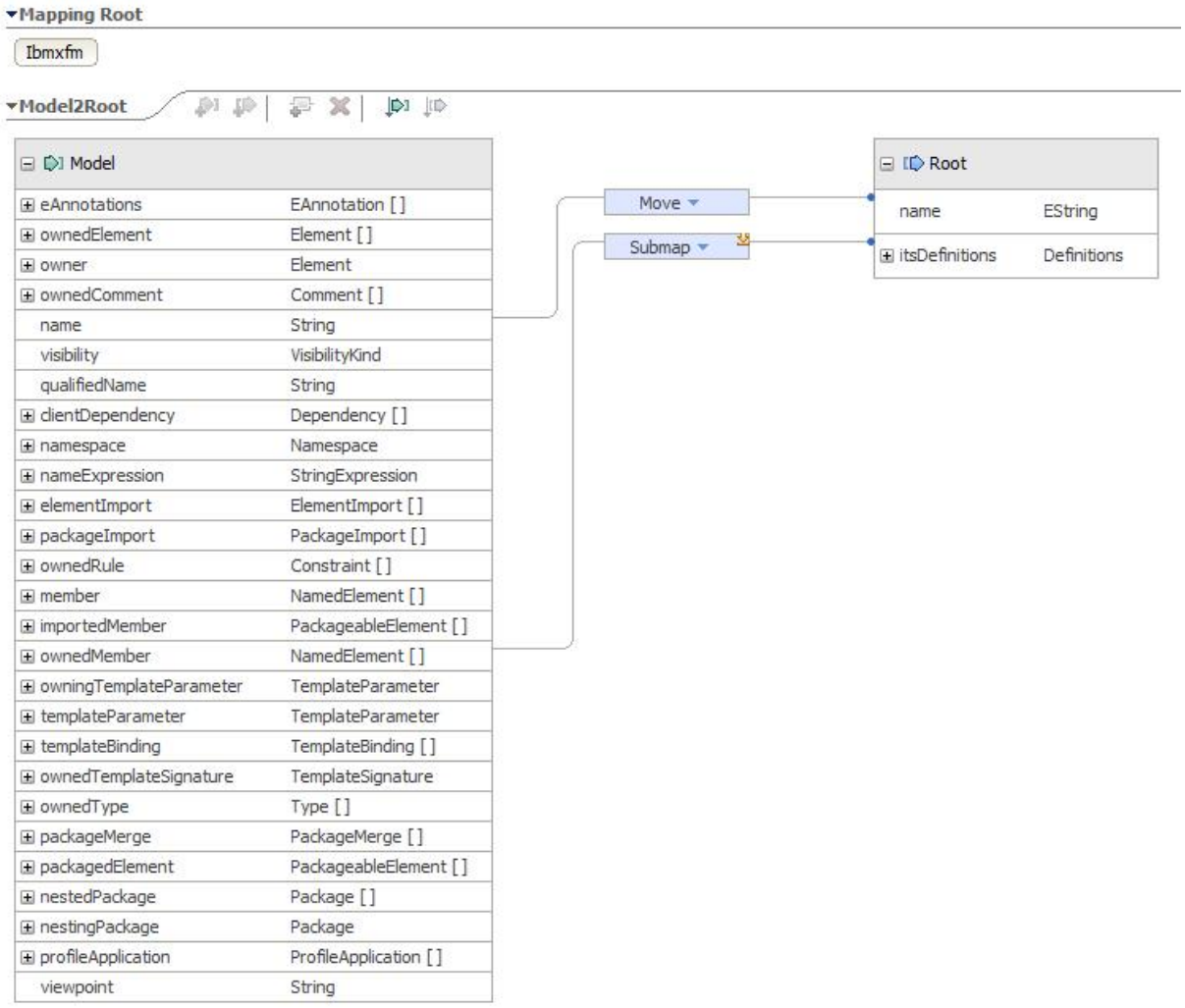
UML Java API, and OCL Java API for Java can be used for writing Java programs to perform M2M and M2T transformations.

4.2 Results from Exploration

The PIM model, and the Platform specific language resulting from the exploration are presented as tree views in Figure 5 and Figure 6 respectively in Appendix A. In Appendix B, the output PSM model resulting from both the M2M transformation technologies (ATL and IBM TF) is presented in Figure 7 as a tree view and textually in XML Metadata Interchange (XMI) format along with the output (C++ code) from the M2T transformation tool (Acceleo). An example transformation rule in IBM TF and ATL are presented in Figure 3 for the PIM, and the Platform Specific Language in Appendix A. An example transformation in Acceleo is provided in Figure 4. We do not present the details of the the implementation (Projects in ATL, IBM TF, and Acceleo) as it is beyond the scope of this paper. However, reflections from the exploration are presented below:

ATL

ATL is textual. The mix imperative and declarative constructs negatively affects learnability, understandability, and suitability, but the mix makes the language powerful and compact. Once the learning barrier is passed, the operability is very high as it allows a user to author code with full control: an advantage. One can use Object Constraint Language (OCL) [OMG, 2010] directly with the constructs of ATL to query model elements: an advantage. If the code has errors causing execution failures, it is very easy to debug as the compiler reports on them, which gives it a better analyzability: an advantage. If the PSM meta-model has been updated, the ATL module senses these changes and reports on them, thus providing positive experiences for changeability: an advantage. The PIM models that were used for transformations had to be exported (a disadvantage) to their plain UML definitions from the IBM Modeling projects (used by the department at Ericsson AB) to be used for transformations. This adds an extra step (a disadvantage) and could be difficult to synchronize source models in case changes have been made in between: a disadvantage. As well,



(a) Visual Editor in IBM TF showing the *mapping* between top-level Model element from PIM to top-level Root element in PSM

```
--Matched Rule
rule Model2Root{
    from umlModel : UML!Model
    to root : PSM!Root (
        name <- umlModel.name,
        itsDefinitions <- umlModel.nestedPackage->collect(x | thisModule.genDefinitions(x)
        )
    )
}
```

(b) Code in ATL with OCL constructs showing transformation rule to transform top-level Model element from PIM to top-level Root element in PSM.

Figure 3: M2M transformation in IBM TF and ATL. Transformation rules in 3a and in 3b produce the same results. The PIM model, and the Platform Specific language are shown in Figures 5 and 6 in Appendix A.

```
[comment encoding = UTF-8 /]
[module generate('/com.ericsson.ex1.IWU.trial.ibm_xfm/m2m/psm.ecore')/]

[template public generate(root: Root)]

    [comment @main /]
    [file (root.name+'DataTypes.hpp', false, 'UTF-8')]
    #ifndef [root.name.toUpper()/]_DATATYPES_HPP
    #define [root.name.toUpper()/]_DATATYPES_HPP

    namespace [root.name/]{

    [for (dc: DataClass | root.itsDefinitions.itsPlaceHolder.itsData)]
        [writeException(dc)/]
    [/for]

    [for (dc: DataClass | root.itsDefinitions.itsPlaceHolder.itsData)]
    [writeDataTypes(dc)/]
    [/for]
    }

    [/file]
```

Figure 4: M2T transformation in Acceleo. The figure presents part of the Acceleo module which produces the output shown on page 17 in Appendix B.

there is no graphical support for the exported UML apart from a tree view: a disadvantage. We had to study the ATL documentation, the OCL specification [OMG, 2010], and perform some examples to write transformations in ATL for the given case.

IBM TF

IBM TF provides a visual editor: a great advantage. Once mapping of objects from PIMs and PSMs is specified visually, the framework allows generation of code in Java. Along with learning the features and functionalities of the visual tool, one has to learn the Java APIs for full control over the transformation: a disadvantage. This affects negatively on the learnability, understandability and operability of the tool. In addition to the generated transformation code, the tool generates the code to run as an Eclipse plugin environment (in a separate package) which affects negatively on analyzability of the generated code: a disadvantage. Changes to the PSM meta-model is not reflected in the visual editor which gives changeability a negative experience: a disadvantage. However, these negative experiences are compensated with suitability of this tool as one has to deal with a single programming paradigm: an advantage. The IBM Modeling projects could be directly fed into transformations without the need to export them: an advantage. We had to study the documentation for IBM TF, UML API for Java, and OCL API for Java in order to write transformations in this framework for the given case.

Acceleo

Acceleo was the single M2T tool used for this study. Documentation and support provided for this tool was extraordinary: an advantage. It required minimum effort to implement and execute M2T transformations, giving an overall positive experience: an advantage. We had to study parts of the documentation and go through some examples to write M2T transformations for the case.

4.3 Results from Evaluation

The results from Evaluation are given in Table 1 and Table 2. Two of the leftmost columns in Table 1 list the qualities and subqualities found. Table 2 defines a generic scale for assessment of all qualities and subqualities.

4.4 Results from Validation

Table 1 is the finalized and updated quality model with the ranking of the transformation technologies after three sessions of presentation and feedback with four participants from the Validation phase (Section 2.2.4) of the study. Answers to the feedback questions are presented in Table 3.

Quality	Sub-characteristics	Transformation Technologies			
		M2M		M2T	
		ATL	IBM Transformation	Acceleo	Java APIs
Usability	Understandability	++	++	+++	+++
	Learnability	- -	++	+++	++
	Operability	+++	++	+++	++
	Overall	+	++	+++	++
Maintainability	Analyzability	++	-	++	-
	Changeability	++	-	++	+
	Overall	++	-	++	-
Functionality	Suitability	++	+++	+++	+
	Accuracy	+++	+++	+++	+++
	Overall	++	+++	+++	++
Scalability	**Overall**	+++	++		+

Table 1: Quality model and ranking for model transformation technologies.

Scale	Explanation
+++	Very positive experience
++	Positive Experience
+	Somewhat positive; improvements can be made
-	Negative experience
- -	Bad experience, but not a fundamental problem; can be fixed by tool vendor or in-house project
- - -	Bad experience; cannot be resolved.

Table 2: Generic scale used for assessment of qualities and sub-qualities of model transformation technologies.

Questions	Answers		
	Participants 1 & 2	Participant 3	Participant 4
1. Do you miss any qualities?	No, the given qualities are good.	No, these qualities are enough.	Yes, Performance and Expandability.
2. Which of these qualities in the quality model are of most importance to Ericsson AB? Could you rank them?	1. Maintainability 2. Usability 3. Functionality 4. Scalability	1. Usability 2. Maintainability 3. Functionality 4. Scalability	1. Functionality 2. Usability
3. Could you also rank the technologies ?	1. IBM TF Framework 2. Acceleo 3. ATL 4. Java API	1. IBM TF Framework 2. Acceleo 3. ATL 4. Java API	Preferred functionality over usability given any tool
4. Why are they ranked this way?	All the participants had similar opinions that the tool should be of high usability (graphical interface for IBM TF framework), have long-term support and have the ability to support extensions to the transforms.		
5. Which technology from model to model transformation (IBM, ATL, Java API) or model to text has the greatest potential from your perspective?	Acceleo and ATL due to its open source nature [EPL, 2011]	Accelo and ATL due to its open source nature [EPL, 2011]	Mature version of IBM TF technology or the one that gains the wider acceptance.
6. Is there anything that you would like to reflect more on Model Transformation?	All the participants agreed that model transformation technology should be applied in practice, although concerns about scalability was expressed.		

Table 3: Feedback Questions and Answers

5 Discussion

We set out to explore available model transformation technologies from an industrial perspective. We found qualities that were important to industry for the particular case, assessed these qualities using our experiences from exploring the technologies and validated our assessment through feedback with MDA engineers working in the industry. We discuss our findings here and relate them to our research questions and research problem.

The qualities and the sub-qualities in Table 1 are those we found to be important for the case (answering RQ1: What qualities of available model transformation technologies are important to MDA engineers in the software industry?). The evaluation scale (Table 2) applied to each of these qualities for each model transformation technology give the opportunity to argue for one technology over the other (answering RQ2: To which degree do available model transformation tools have these qualities?). Table 3 provides insights on the quality model, evaluation scale, and the model transformation technologies from users point of view (answering RQ3: Which of the found qualities are most important to MDA users in industry?). The choice of a single technology here is a tradeoff. Answers to Question 2 from Table 3 (Which of these qualities in the quality model are of most importance to Ericsson AB? Could you rank them?) is important while making the choice of a technology for any particular case. For our case, we can see that each participant answered differently, although usability ranked highest followed by either functionality or maintainability.

5.1 Rationale for the choice of model transformation technology

5.1.1 Java API is not the choice

Although the Java API approach accounts for better understandability, it is clearly seen from the results of the Exploration and the feedback received during Validation (Tables 1 and 3) that advantages of M2M and M2T tools outweigh the advantages of Java APIs for model transformations.

Furthermore, the participants acknowledged that the guidance provided by M2M and M2T technologies, for solving model transformation problems, is a significant advantage over just using Java APIs (which gives no such guidance), thus M2M and M2T technologies are preferable to Java in practice (question 6 in Table 3).

This excludes Java API as a candidate technology for both M2M and M2T transformation problems.

5.1.2 ATL is preferable over IBM TF

M2M technologies against each other, we appreciated the mixed programming paradigm in ATL (which supported compact programming), but also the visual interface of the IBM TF (which was intuitive and easy to work with). However, taking learnability and maintainability into account, ATL seems to outscore IBM TF.

Learnability
Engineers within the department had strong preference for IBM TF, thanks to its use of the Java API, which the engineers had used extensively, and thus were familiar with and liked.

From a learnability perspective, IBM TF does not require a paradigm shift, which however ATL does (for these engineers). For ATL, learnability is hampered by two factors: documentation is hard to read; the mixed programming paradigm is non-intuitive. However, these limitations are not serious: documentation can be fixed either by an in-house project or by contacting the vendor directly for support; the mixed programming paradigm can be learned by a little training.

Maintainability
The engineers ranked maintainability as the second prioritized quality for M2M technology (question 2 in Table 3).

From Table 1, it is clear that transformations in IBM TF are hard to maintain compared to ATL transformations (analysability and changeability). Fixing these issues require an entirely new architecture for the generated code, which can only be done by the vendor (IBM) if it can be done at all. This makes ATL a better choice in terms of maintainability (given that developers have the sufficient training to work with ATL).

5.1.3 Accelelo is the choice for M2T

For the same reasons, Acceleo is the preferred choice for M2T transformations: usability and maintainability is better than for Java APIs. (Again, modulo some minor initial training.)

6 Related Work

Past studies of model transformation is not new. Mens et al. [2005] in their summary of discussions in a seminar recognize characteristics, success criteria, quality requirements, and programming paradigms

of model transformation tools. Within their success criteria, they present important functional requirements of a model transformation tool or a language. Within quality requirements, they suggest usability, suitability, and scalability as the key quality requirements within model transformation tools. They also suggest that accessing models by means of API's could be an advantage since the programmers will not require extra training. They conclude by saying that a model transformation technology should be selected based on its application domain, which is what our case study provides specific methods for.

Van Gorp and Eshuis [2010] addresses model transformation from the perspective of programming paradigms. The study done in 2010 specifies that the the study of the strengths and weaknesses of transformation approaches across community boundaries (compiler community, program transformation community and business process management community) is relatively new. They also support finding strengths and weaknesses of approaches to transformation for particular application domains. In their study, they report on strengths and weaknesses of a transformation program written in Java compared to a rule based program (GrGen) in the domain of business process management. They find that rule based languages are more appropriate for model transformations for that domain. They address aspects of model transformations such as runtime performance, and code optimization. Our study supplements their research work by providing further quality aspects of model transformation technologies.

Sendall and Kozaczynski [2003] also studied desirable characteristics that a model transformation language should have. They find preconditions, composition, form, and usability to be desirable characteristics of model transformation languages. They also suggest that a transformation language should be able to support the approach Object Management Group is trying to standardize (the MDA approach), which confirms our findings: MDA tools (ATL, Acceleo and IBM TF) are preferred over using general purpose programming languages for model transformations (Java APIs).

7 Conclusion

We set out to explore model transformation technologies in an industrial setting. Our research questions were:
RQ1: What qualities of available model transformation technologies are important to MDA engineers in software industries?
RQ2: To which degree do available model transformation tools have these qualities?
RQ3: Which of the found qualities are most important to MDA users in industry?
In our Exploration activity (Section 2.2.2), we answered our RQ1 by finding important qualities for the explored model transformation technologies. We ranked each of the technologies against the found qualities, answering our RQ2 during Evaluation (Section 2.2.3). In our Validation activity (Section 2.2.4), we validated and updated these assessments by the feedback received from the users for this particular case. The feedback gave us qualities that were important to these users and answered our RQ3 for a particular context within Ericsson AB. By answering the research questions, we were able to provide rationales for the choice of technology for our industrial case. Our findings contribute to the area of model transformations where we confirm that methods for systematic selection of model transformation technologies are needed, and we provide concrete guidance on which qualities of model transformation technologies to investigate, and how to investigate these in an industrial setting. The method designed for this study can be applied for similar evaluation in different application domains and for different model transformation technologies.

Acknowledgments

I would like to thank Lars Pareto for his constant supervision, advice, and direction over the course of my research. I would like to express my gratitude to Staffan Ehnebom, Peter Eriksson R., and Henric Stenhoff at Ericsson AB for their guidance and support without which this study would not have been possible. I would also like to thank Helena Holmström, Carl Magnus Olsson, Agneta Nilsson, Miroslaw Staron and the engineers at Ericsson AB for their valuable time and feedback throughout this research. Last but not least, I would like to thank my parents, K. M. Azizul Bari and Meher Aziz for providing me the opportunity to be where I am and supporting me throughout my life.

References

Software engineering-Product quality-Part1: Quality Model. ISO/IEC 9126-1, 2001.

Eclipse public license, May 2011. URL `http://www.eclipse.org/legal/epl-v10.html`.

I. Benbasat, D. K. Goldstein, and M. Mead. The case research strategy in studies of information systems. *MIS Quarterly*, 11(3):pp. 369–386, 1987. ISSN 02767783. URL `http://www.jstor.org/stable/248684`.

J. W. Creswell. *Research design: Qualitative, quantitative and mixed methods approaches*. Sage Publications, 3rd edition, 2009.

S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. Selecting empirical methods for software engineering research. In F. Shull, J. Singer, and D. I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer London, 2008. ISBN 978-1-84800-044-5. URL http://dx.doi.org/10.1007/978-1-84800-044-5_11. $10.1007/978 - 1 - 84800 - 044 - 5_11$.

Eclipse. Ocl java api, May 2011a. URL http://www.eclipse.org/modeling/mdt/?project=ocl#ocl.

Eclipse. Uml java api, May 2011b. URL http://wiki.eclipse.org/MDT-UML2.

IBM. Ibm transformation framework, May 2011. URL http://publib.boulder.ibm.com/infocenter/rsahelp/v8/index.jsp?topic=/com.ibm.xtools.transform.authoring.doc/topics/ttransm2mover.html.

F. Jouault and I. Kurtev. Transforming models with atl. In J.-M. Bruel, editor, *Satellite Events at the MoDELS 2005 Conference*, volume 3844 of *Lecture Notes in Computer Science*, pages 128–138. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/11663430_14. $10.1007/11663430_14$.

A. Kleppe, J. Warmer, and W. Bast. *MDA Explained, The Model-Driven Architecture: Practice and Promise*. Addison Wesley, 2003.

A. I. . LINA. Atlas transformation language, May 2011. URL http://wiki.eclipse.org/ATL/User_Guide.

T. Mens, K. Czarnecki, and P. V. Gorp. 04101 discussion – a taxonomy
of model transformations. In J. Bezivin and R. Heckel, editors, *Language Engineering for Model-Driven Software Development*, number 04101 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. URL http://drops.dagstuhl.de/opus/volltexte/2005/11.

M. J. Miller, J. Mda guide. 2004, 2003.

Obeo. Acceleo, May 2011. URL http://www.acceleo.org/pages/home/en.

OMG. Object constraint language, 2010. URL http://www.omg.org/spec/OCL/2.2/PDF/.

D. Pilone and N. Pitman. *UML 2.0 in a Nutshell*. O'Reilly Media, July 2005.

P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14: 131–164, 2009. ISSN 1382-3256. URL http://dx.doi.org/10.1007/s10664-008-9102-8. 10.1007/s10664-008-9102-8.

B. Selic, K. Hussey, and T. McClean. An extended survey of open source model-based engineering tools, May 2010. URL http://wiki.eclipse.org/images/d/dc/Report.external.bvs.pdf.

S. Sendall and W. Kozaczynski. Model transformation: the heart and soul of model-driven software development. *Software, IEEE*, 20(5):42 – 45, Sept.-Oct. 2003. ISSN 0740-7459. doi: 10.1109/MS.2003.1231150.

P. Van Gorp and R. Eshuis. Transforming process models:
executable rewrite rules versus a formalized java program. In D. Petriu, N. Rouquette, and Ø. Haugen, editors, *Model Driven Engineering Languages and Systems*, volume 6395 of *Lecture Notes in Computer Science*, pages 258–272. Springer Berlin / Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-16129-2_19. $10.1007/978 - 3 - 642 - 16129 - 2_19$.

R. K. Yin. *Case Study Research Design and Methods*. Sage Publications, 3rd edition, 2003. Applied Social Research Methods Series, Vol 5.

A Models and Meta-models used as input

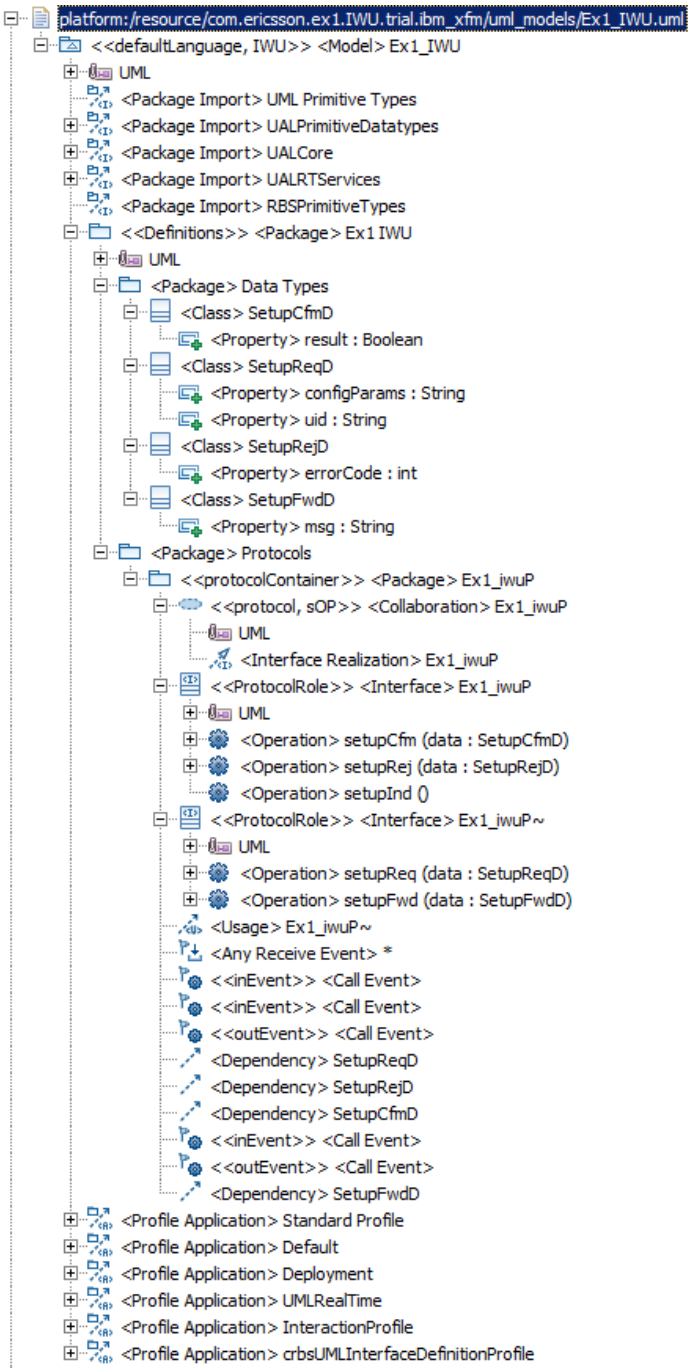


Figure 5: The figure presents the PIM model used as an example source in the study.

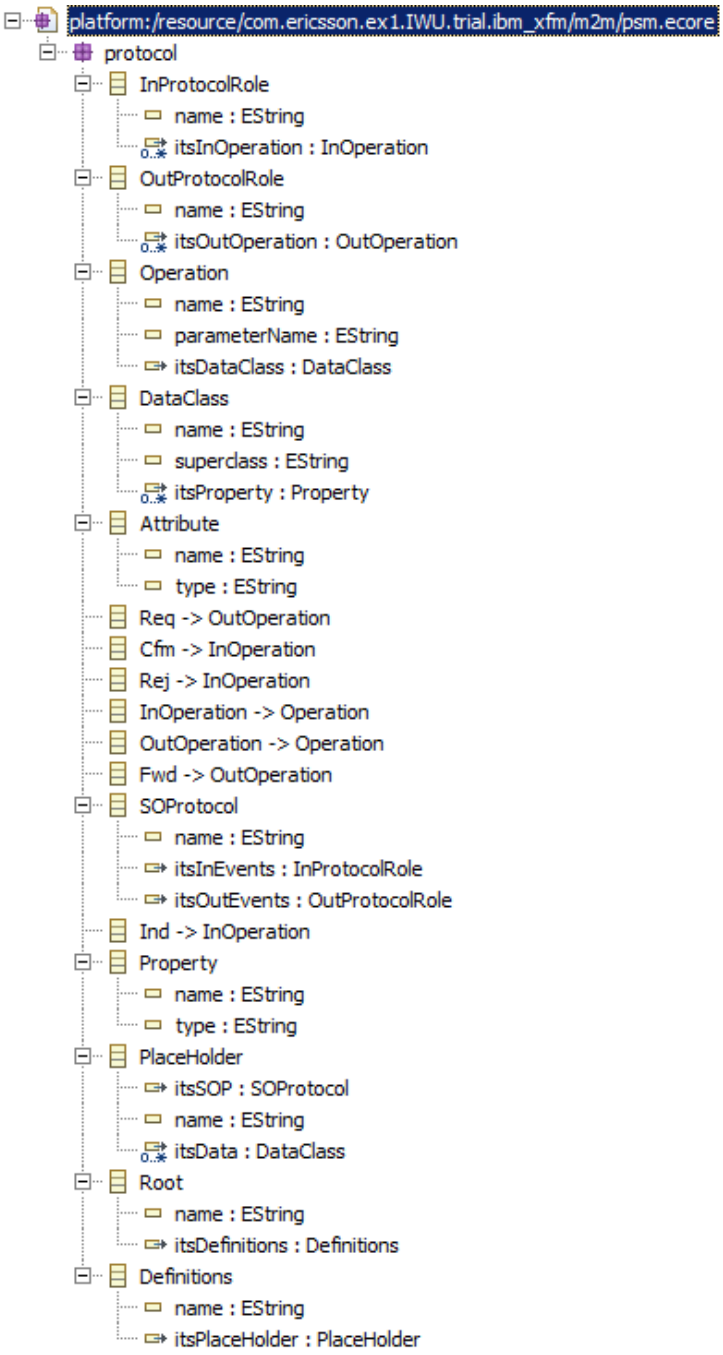


Figure 6: The figure presents the Platform Specific Language developed for the purpose of this study.

B Output from the technologies

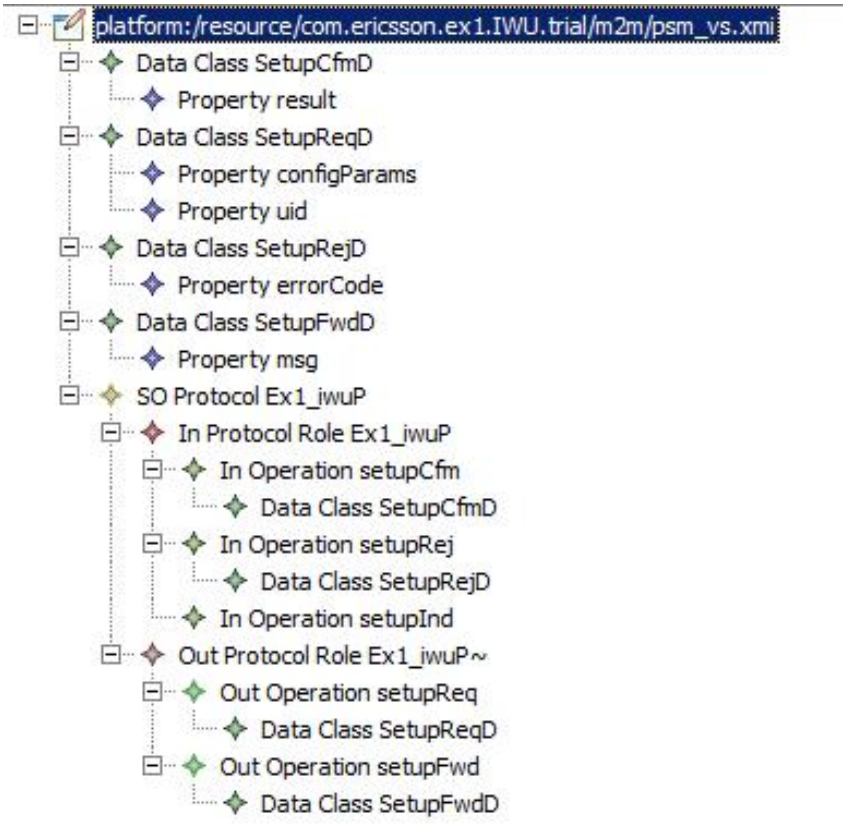


Figure 7: The figure presents the PSM model produced by the M2M tools.

The XMI representation of the PSM model is provided below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI" xmlns:protocol="http://protocol/1.0">
  <protocol:DataClass name="SetupCfmD">
    <itsProperty name="result" type="Boolean"/>
  </protocol:DataClass>
  <protocol:DataClass name="SetupReqD">
    <itsProperty name="configParams" type="String"/>
    <itsProperty name="uid" type="String"/>
  </protocol:DataClass>
  <protocol:DataClass name="SetupRejD">
    <itsProperty name="errorCode" type="int"/>
  </protocol:DataClass>
  <protocol:DataClass name="SetupFwdD">
    <itsProperty name="msg" type="String"/>
  </protocol:DataClass>
  <protocol:SOProtocol name="Ex1_iwuP">
    <itsInEvents name="Ex1_iwuP">
      <itsInOperation name="setupCfm" parameterName="data">
        <itsDataClass name="SetupCfmD"/>
      </itsInOperation>
      <itsInOperation name="setupRej" parameterName="data">
        <itsDataClass name="SetupRejD"/>
      </itsInOperation>
      <itsInOperation name="setupInd"/>
    </itsInEvents>
    <itsOutEvents name="Ex1_iwuP~">
      <itsOutOperation name="setupReq" parameterName="data">
        <itsDataClass name="SetupReqD"/>
      </itsOutOperation>
      <itsOutOperation name="setupFwd" parameterName="data">
        <itsDataClass name="SetupFwdD"/>
      </itsOutOperation>
    </itsOutEvents>
  </protocol:SOProtocol>
</xmi:XMI>
```

The C++ source code provided below was generated by the M2T transformation tool, Acceleo, from the PSM model (the code presented below is for DataClasses only):

```
#ifndef Ex1_IWU_DATATYPES_HPP_
#define Ex1_IWU_DATATYPES_HPP_
namespace Ex1_IWU
{
    /* ExceptionS */
    struct ExceptionS{
        int errorCode
    };
    /*SetupCfmD*/
    struct SetupCfmD{
        Boolean result
    };
    /*SetupReqD*/
    struct SetupReqD{
        String configParams
        String uid
    };
    /*SetupRejD*/
    struct SetupRejD{
        int errorCode
    };
    /*SetupFwdD*/
    struct SetupFwdD{
        String msg
    };
}
#endif /* Ex1_IWU_DATATYPES_HPP_ */
```