Towards Evaluating Maintainability Within Model-Driven Environments*

Thomas Goldschmidt, Jens Kübler
FZI Forschungszentrum Informatik
Karlsruhe, Germany
goldschmidt, kuebler@fzi.de

Abstract: Model Driven Software Development (MDSD) has matured over the last few years and is now becoming an established technology. One advantage that is promoted by the MDSD community is the improved maintainability during the systems evolution over conventional development approaches. Compared to code-based development (meta-)models and transformations need to be handled differently when it comes to maintainability assessments. However, a comprehensive analysis of the impact of the model-driven development approach on the maintainability of a software system is still lacking. This paper presents work towards the finding of appropriate approaches and metrics for measuring the maintainability and evolution capabilities of artefacts within model-driven environments. We present our first steps and further ideas on how to tackle this problem.

1 Introduction

Maintainability metrics for object oriented (OO) systems have been around for several years now and are extensively explored [LH93]. However, with the advent of model-driven software development (MDSD)[VS06] a new area emerges where maintainability also needs to be assessed. Maintenance is one of the major cost factors in software development [Mas05]. MDSD claims to provide aid in reducing these costs. Though, currently it still needs to be validated whether MDSD really improves the maintainability of a system. One step that is needed to fulfil this issue is the identification of comprehensive, comparable and automatically collectable metrics for this paradigm.

For models and transformations the conventional code metrics can not be applied to obtain meaningful results. For example, it does not make sense to take the lines of code metric of generated code into account to measure its maintainability. First steps were made on how to assess maintainability within UML models by partly adopting metrics from object oriented development [MSZ$^+$04] and employing new UML specific metrics [XKL04]. It remains unclear how these metrics can be generalized and also utilized to measure the maintainability of domain specific languages (either based on UML profiles

*This work is part of the research project *Model Driven Integration of Business Information Systems (MINT)* [SSMG06]. MINT is supported by the German *Federal Ministry of Education and Research* in the scope of the *Forschungsoffensive Software Engineering 2006.*

or own metamodels). Additionally changes that are made to the transformations in such an environment also need to be taken into account. Another important aspect that needs to be accounted for is the comparability of the metrics to those of non-model-driven approaches to decide if and in which scenarios such an approach is beneficial to maintainability costs.

Based on a case study that utilizes all different MDSD concepts, such as profiles, metamodels, model to model and model to code transformations we will set up a goal, question, metrics (GQM [BCR94]) plan to identify potential metrics.

The structure of this paper is as follows. First, we briefly discuss related work and indicate some shortcomings of existing approaches for maintainability assessment. In section 3, we present our ideas on how the maintainability can be measured. Section 4 describes the setup of our case study which will be used during further development of our approach. We finish with an overview on the current status of our efforts and a concluding summary in section 5 and 6.

2 Related Work

Maintainability of object oriented systems from an UML perspective with respect to metrics has been investigated in several works ([MP07],[MSZ$^+$04],[XKL04] and evaluated in [SMK02] such as the number of classes, attributes or number of generalisations. However, the metrics are founded solely on models which are based on the UML metamodel and do not cover the application of profiles to these models, other metamodels and transformations. Preliminary work has been done on how complex semantic constraints formulated in OCL affect maintainability in [RGPM05]. As OCL is a normative reference for the Query/View/Transformation specification (QVT) [Obj] these findings may be included in analysis to transformations as well. In one of our previous case studies we already identified some scenario based metrics using the GQM approach that allow for the comparison of maintainability of model-driven and non-model-driven techniques [GWR07, GWR08]. Unfortunately these metrics are only partially generalizable for our purpose as they are specific to the given scenario.

3 Ideas for the Assessment of Maintainability in a Model-driven Environment

Several definitions of maintainability have been proposed i.e. in IEEE 610.12 or ISO/IEC 12207 and 14764. The work of [OH92] splits software maintainability into three main aspects : (1) The management practices, (2) the operational hardware and software environment that is involved with the software system under maintenance, and (3) the target software system itself. Instead of tampering with a precise definition we tend to use a constructive approach that focuses on actions necessary for maintenance. In the broader sense of software quality ISO 9126 defines a quality model which is therfore our source of inspiration for a GQM plan. To address maintainability in this context we analyse the general structure of the model-driven paradigm which consists of (meta-)models and transformations. Therefore we propose to establish detailed metrics on these two components through our GQM plan.

3.1 Metrics on Models

Starting with models we recognize a separation of domain aspects from functional aspects which can be found i.e. in [BGeA]. This can be implemented in various ways such as applying different profiles (that is stereotypes and tagged values) to annotate functional information to the platform independent models or keeping two separate models which are woven together. On one hand structural metrics for both aspects should be straight forward to define on the other hand we regard semantics like OCL expressions more complicated to measure. Suppose a functional aspect "persistent" on class "A" which would imply the semantic constraint, that all properties of that class are to be persistent. At first glance it is unclear up to which degree of abstraction semantic constraints are more maintainable than explicit syntactic constructs. In this case verbosity conflicts with conciseness as described in [MVG05]. We consider an unified model of metrics for domain and functional aspects useful, if it is parametrizable so that for each aspect classification of good and bad design is possible.

3.2 Metrics on Transformations

To cover maintainability on transformations two approaches are considered. The first approach is based upon model comparison and similarity between original and transformed models which is suitable for M2M transformations. Similarity is an indicator for complexity of transformations and hence it relates to the maintainability of the transformation. [Ruf03] gives a good overview and classification how model similarity can be addressed.

The second approach focuses on transformations themselves. [CH03] outlines several transformation features that may be used to identify components of a transformation that are crucial to its maintainability. Regardless of the transformation technique used, i.e. declarative or imperative, transformations can be treated as rule systems with rules consisting of a left hand side and a right hand side . Traceability of rules is considered an important feature of transformations as it connects source and target models which offers impact analysis that is analysis of subsequent changes caused by an initial change. [CH06]. Therefore, we think support for traceability by the transformation engine influences the maintainability of a transformation substantially. Further transformation features are considered feasible such as rule organisation which covers reuse and modularity mechanisms that are outlined in [CH06]. [KvdBJ06] introduces the notion of patterns with respect to transformations. Such patterns may affect maintainability metrics both in a harmful and leveraging way and should be carefully inspected. Suppose a façade pattern in a conventional code centric software project that accumulates several methods from various parts of the application. Although more maintainable this pattern may result in poor cohesion values which indicate bad maintainability.

3.3 Measurement Process

To extract meaningful metrics for maintainability from the numerous metrics available we aim to setup a GQM plan [BCR94]. Through the goal driven definition of the metrics according to this plan we are able to directly draw conclusions from our gathered metrics. A question we consider relevant is which changes in the source or target model cause what amount of adoption in the transformation. Switching from one platform specific model to another one is often stated as a feature of model driven software development so we plan to analyse the effort of switching models.

Integrating the measurement process into the transformation is a significant issue to tackle as it supports the engineer with useful information during the development process. Therefore we need to rely on openness and extendabilty of the tools we consider to use.

4 The Case Study

To be able to ensure the comparability of our metrics and also to identify the possible interdependencies between the different aspects in a model-driven environment we are currently developing a case study based on the domain of persistency frameworks. Within this case study we will use a standard Model Driven Architecture (MDA) process as proposed by the Object Management Group (OMG) in [Obj03]. This process includes the definition of platform independent and platform specific models as well as transformations to models and code.

On the one hand we chose the domain of persistency frameworks because there are many different frameworks for this purpose that can serve as platform specific target model for the model transformation and on the other hand because there are several platform independent functional aspects that are to be handled by a model transformation to a PSM. The transformation rules should reveal themselves as basic which enables easy analysis of frequently arising transformation structures.

In our concrete study we use UML to model the platform independent model, which allows us to re-use, or at least directly compare existing model metrics and decide whether they are usable in a broader sense. Furthermore we use the profiling mechanism of UML to create profiles for important aspects within our target domain, such as object relational mapping, object identity or validation. This profile will be kept platform independent. As an example for the platform specific model we will use the Java Persistence API (JPA) part of the EJB3 specification [jsr].

To define the profiles and create the necessary models we plan to use MagicDraw UML [mag] as it provides extended support for both UML models and profiles. We intend to

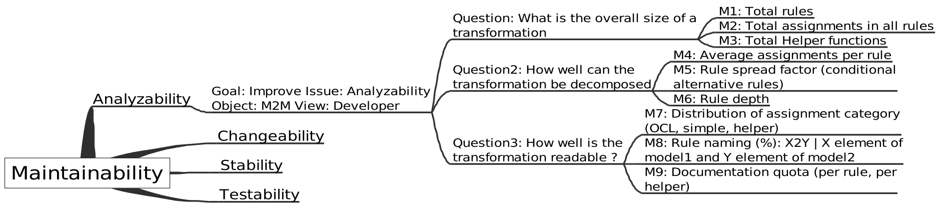
Maintainability
Analyzability
Changeability
Stability
Testability
Goal: Improve Issue: Analyzability
Object: M2M View: Developer
Question: What is the overall size of a transformation
Question2: How well can the transformation be decomposed
Question3: How well is the transformation readable ?
M1: Total rules
M2: Total assignments in all rules
M3: Total Helper functions
M4: Average assignments per rule
M5: Rule spread factor (conditional alternative rules)
M6: Rule depth
M7: Distribution of assignment category (OCL, simple, helper)
M8: Rule naming (%): X2Y | X element of model1 and Y element of model2
M9: Documentation quota (per rule, per helper)

Figure 1: Snapshot of Goal Question Metric Plan for Analysability in M2M Transformations

focus our efforts for M2M transformations on the relational part of the QVT specification where mediniQVT [med] recently became available as an implementation. For M2T transformations we intend to use openArchitectureWare (OAW) [oaw] as it is a quite mature and well established framework with templating, modularization and aspect weaving capabilities.

Once usable metrics are found applying those to legacy software systems will be subject to our research so that a real world example is provided.

5 Current Status

We defined a PIM profile for persistence, a PSM profile for JPA and implemented a model to text transformation which generates the appropriate Java classes. We are yet to implement a M2M transformation which should provide an “easy” example for a transformation. As measurement process plan we set up a GQM plan which currently covers analysability. We identified metrics for models, stereotypes, differences between models as well as metrics for M2M transformations which are partly shown in figure 1. M2T metrics are not considered at this time but may be transferred from our M2M findings in future research. Our metrics application is implemented as a Eclipse plugin using OCL as query language for models. As many constituents of the MDA approach are available as Ecore models such as OCL, QVT or UML, we have flexible measurement capabilities at hand. Upcoming investigation will tackle classification and aggregation of measures and the issue of how to establish validity of our findings.

6 Conclusion

In this paper we presented our ideas on how to approach the problem of evaluating maintainability within a model-driven development environment. We identified that not only the maintainability of the developed domain models but also the metamodels, profiles and transformations that are used have a great influence on the effort of maintaining such a system. We will use a GQM plan applied within a case study to find these metrics.

References

[BCR94] V. Basili, G. Caldeira, and H. D. Rombach. *Encyclopedia of Software Engineering*, chapter The Goal Question Metric Approach. Wiley, 1994.

[BGeA] Krishnakumar Balasubraman, Aniruddha Gokhale, and et Al. Weaving Deployment Aspects Into Domain-Specific Models. last access: 20.12.2007.

[CH03] Krzysztof Czarnecki and Simon Helsen. Classification of Model Transformation Approaches. In *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 2003.

[CH06] K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45(3):621–645, July 2006.

[GWR07] T. Goldschmidt, J. Winzen, and R. Reussner. Evaluation of Maintainability of Model-driven Persistency Techniques. In *IEEE CSMR 07 - Workshop on Model-Driven Software Evolution (MoDSE2007)*, 2007.

[GWR08] T. Goldschmidt, J. Winzen, and R. Reussner. A Case Study Evaluation of Maintainability and Performance of Persistency Techniques. In *Proceedings of the 30th international conference on Software engineering ICSE '08*, 2008. to appear.

[jsr] JSR 220: Enterprise JavaBeans 3.0. last access: 20-12-2007.

[KvdBJ06] Ivan Kurtev, Klaas van den Berg, and Frédéric Jouault. Evaluation of Rule-based Modularization in Model Transformation Languages illustrated with ATL. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 06)*, pages 1202–1209, Dijon, France, 2006. ACM Press.

[LH93] Wei Li and Sallie Henry. Object-oriented metrics that predict maintainability. *J. Syst. Softw.*, 23(2):111–122, 1993.

[mag] MagicDraw UML. `http://www.magicdraw.com/`. Last Access: 20-12-2007.

[Mas05] Dieter Masak. *Legacysoftware*. Springer, 2005.

[med] ikv++ mediniQVT. last access: 20-12-2007.

[MP07] Jacqueline A. McQuillan and James F. Power. On the Application of Software Metrics to UML Models. In *Models in Software Engineering*, Lecture Notes in Computer Science, pages 217–226. Springer, 2007.

[MSZ$^+$04] Haohai Ma, Weizhong Shao, Lu Zhang, Zhiyi Ma, and Yanbing Jiang. Applying OO Metrics to Assess UML Meta-models. In *UML 2004 - The Unified Modelling Language: Modelling Languages and Applications. 7th International Conference, Lisbon, Portugal, October 11-15, 2004. Proceedings*, volume 3273 of *Lecture Notes in Computer Science*. Springer, 2004.

[MVG05] Tom Mens and Pieter Van Gorp. A taxonomy of model transformation. In *Proc. Int'l Workshop on Graph and Model Transformation*, 2005.

[oaw] openArchitectureWare. `http://www.openarchitectureware.org/`. Last Access: 20-12-2007.

[Obj] Object Management Group. MOF QVT Final Adopted Specification. http://www.omg.org/cgi-bin/doc?ptc/05-11-01.pdf.

[Obj03] Object Management Group, Framingham, Massachusetts. *MDA Guide Version 1.0.1*, June 2003.

[OH92] P. Oamn and J. Hagemeister. Metrics for assessing a software system's maintainability. In *Software Maintenance, 1992. Proceerdings., Conference on*, 1992.

[RGPM05] L. Reynoso, M. Genero, M. Piattini, and E. Manso. Assessing the impact of coupling on the understandability and modifiability of OCL expressions within UML/OCL combined models. *Software Metrics, 2005. 11th IEEE International Symposium*, pages 10 pp.–, 19-22 Sept. 2005.

[Ruf03] Raimi A. Rufai. New Structural Similarity Metrics for UML Models. Master's thesis, King Fahd University of Petroleum and Minerals, Dharan 31261, Saudi Arabia, January 2003.

[SMK02] Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi, editors. *Conceptual Modeling - ER 2002, 21st International Conference on Conceptual Modeling, Tampere, Finland, October 7-11, 2002, Proceedings*, volume 2503 of *Lecture Notes in Computer Science*. Springer, 2002.

[SSMG06] Niels Streekmann, Ulrike Steffens, Claus Möbus, and Hilke Garbe. Model-Driven Integration of Business Information Systems. In *Softwaretechnik-Trends*, volume 26, pages 9–13, November 2006.

[VS06] Markus Völter and Thomas Stahl. *Model-Driven Software Development*. Wiley, 2006.

[XKL04] Baowen Xu, Dazhou Kang, and Jianjiang Lu. A Structural Complexity Measure for UML Class Diagrams. In *Computational Science - ICCS 2004, 4th International Conference, Kraków, Poland, June 6-9, 2004. Proceedings*, volume 3036 of *Lecture Notes in Computer Science*. Springer, 2004.