A Domain Specific Transformation Language to Support the Interactive Definition of Model Transformation Rules

Luis Silvestre

Computer Science Department, University of Chile
Beauchef 851, 8370459 Santiago, Chile
lsilvest@dcc.uchile.cl
http://www.dcc.uchile.cl

Abstract. Model-driven engineering (MDE) is a new software development paradigm that intends to improve software construction by raising the abstraction level through the use of models and transformations. Recently, MDE has also been used for automatic change evolution and in particular for tailoring software processes. Even though there are some proposals for automatically generating part of the transformation models, they are not easily applicable in the software industry because the potential users usually do not have the capabilities of developing or maintaining transformations. Proposals trying to address this problem should balance the formality required by MDE and the usability needed by the users. This research applies higher-order transformations (HOTs) and decision models for generating transformations to tailor software process models with no direct user interaction with the code. To that end, a graphical interface was defined with such a purpose, so that transformation rules are interactively defined using a domain specific languaje (DSL), and the tailoring transformation is automatically generated and applied using a domain specific transformation language (DSTL).

Keywords: Model-driven Engineering, Software Process Tailoring, Domain Specific Transformation Language

1 Introduction

Model-driven engineering (MDE) [23] looks forward to improving development productivity and software quality by reducing the semantic gap between the problem domain and its solution, using models and transformations that raise the abstraction level of the managed concepts [4]. MDE has been applied in the development of applications [2], automatic change evolution [13] and also in software process engineering [7]. Recently, MDE techniques have also been applied for tailoring software process models [18]. In Hurtado et al. [14, 15], two source models are considered as input for the tailoring transformation: (1) *the organizational process model* of a certain company based on SPEM (Software and Systems Process Engineering Metamodel) [20], and (2) *the concrete context model* for a project where the process needs to be applied. The *tailoring transformation* is implemented using ATL [6]. Although such a proposal proves that the approach is technically feasible, there are still factors that make the solution difficult to deploy in software companies, and more specifically in small software enterprises (SSEs). First, defining and maintaining tailoring transformations with two source models is in itself a complex task. Second, the tailoring knowledge encapsulated within the transformation rules is generally only managed by the company's process engineer, but he usually is not familiar with the syntax and semantics of transformation languages. Therefore, he could hardly develop and maintain the transformation code.

As a means to address such a problem, this research proposes the automatic generation of tailoring transformations through the interactive definition of tailoring rules, that improves the usability and hides the complexity of MDE components. In our research, a graphical environment allows rule definition as a variation decision model (VDM). We developed a HOT [30] that takes the VDM as input and generates a tailoring transformation written in a DSTL (in this case, a subset of ATL). Defining the transformation rules using our DSL is easier than writing them directly in a transformation language such as ATL [6] or QVT [21], without sacrificing expressiveness for the application domain.

2 Problem Statement

Tailoring transformations implementation is difficult because it requires high expertise about MDE concepts and also about the application domain, and these two kinds of knowledge are almost never mastered by the same person. MDE knowledge can be obtained through training, but highly skilled people are still required, and these people are not generally in charge of the process in SSEs.

MDE solutions are powerful and suitable for different application domains, but they are almost always complex. There are social and organizational factors that threaten industrial adoption [16] (e.g., technological factors, resistance to change). Probably, if we could lower complexity, then adoption would be higher. In this sense, if SSEs would like to reuse their processes through model-based tailoring by defining and applying the transformation rules, they should not require a direct interaction with the source code of any model transformation.

3 Related Work

There have been several proposals to deal with the difficulty of writing transformations. Some high abstraction level transformation languages specifically created for this purpose have been proposed, such as ATL [6] or QVT [21]. Provided that these languages have very specific syntax and semantics, few people in SSEs have the capabilities of developing or maintaining this kind of code.

Varró and Balogh, through the VIATRA framework [31], provide an integrated environment for building and executing transformations at an even higher abstraction level. It includes its own language that intends to hide the complexity of transformations and is supported by Eclipse. However it does not provide an easy-to-use environment for process engineers.

Provided that transformations can also be considered as models [5], HOTs [30] are a special kind of transformations that may take a transformation as input and/or generate a transformation as output. In particular, HOTs that generate transformations are called synthesis [29]. This is the kind of HOTs in which we are interested.

The Atlas Model Weaver (AMW) [10] tool includes an interactive interface for defining a weaving model that defines the relationships between two or more models. The weaving model can then be used as input for automatically generating model transformations. The purpose of AMW is to generate transformations for traceability or matching, so the rules are simple and they do not include complex structures; in particular it is not possible to include conditions for matching elements as we need for tailoring process models according to the project context. Nevertheless, we follow the structure of the AMW tool for our solution: defining the relationship between both input models and the output model, and use this model as the input for a HOT.

There are other MDE proposals that intend to address usability. MOLA [17] and GREaT [1] allow specifying transformations through visual mapping patterns. They specify rules and mappings using class diagrams, but considering an environment inspired in activity diagrams. Both works define the possibility of establishing relationships between metamodel attributes and elements. But they still need the user to directly interact with metamodels and class diagrams, which still represents a strong restriction for process engineers in terms of usability.

There are some recent proposals such as MTBE (Model Transformations By Example) [33] and MTBD (Model Transformation By Demonstration) [28] that present innovative solutions for simplifying the implementation of model transformations using visual strategies and patterns. These strategies generate part of the code of the model transformations; however, it is still needed to complete such code. Therefore, this represents a semi-automatic approach to generate model transformations that is still not enough for process engineers.

4 Proposed Solution

The proposed solution consists of applying diverse MDE concepts to allow automatic transformation generation in a usable way for final users. The generated transformations should be *expressive* enough for its purpose, i.e. for tailoring software process, the interactive definition of transformations should be *usable* for target users, i.e. process engineers, and the complete generation process should be transparent, and no direct interaction with the code should be necessary. This proposal focuses particularly on the need of SSEs, which have to adapt their software processes for efficient development, but they generally do not count on highly specialized people for manually tailoring their processes. The proposed solution is called Architect of Tailoring Rules (ATR). ATR is a

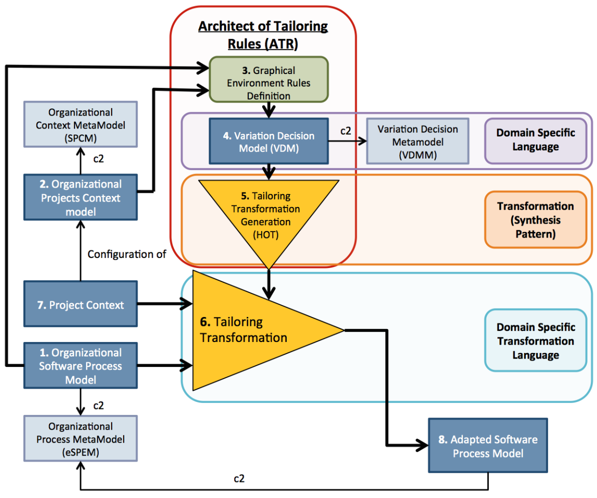

Architect of Tailoring Rules (ATR)
3. Graphical Environment Rules Definition
Organizational Context MetaModel (SPCM)
c2
2. Organizational Projects Context model
4. Variation Decision Model (VDM)
Variation Decision Metamodel (VDMM)
Domain Specific Language
5. Tailoring Transformation Generation (HOT)
Transformation (Synthesis Pattern)
Configuration of
7. Project Context
6. Tailoring Transformation
Domain Specific Transformation Language
1. Organizational Software Process Model
Organizational Process MetaModel (eSPEM)
8. Adapted Software Process Model
c2

Fig. 1. Architecture of the Proposed Solution

model-based tool that allows process engineers to interactively define the transformation rules using a graphical environment. The output of the tool is the tailoring transformation that can be used to adapt the organizational software process.

The proposed solution has four major components:

– *The graphical environment rules definition* requires two source models. The first input is the *organizational software process model* defined by the SSE for guiding its software developments (component 1 in Fig. 1). This model contains all activities, roles, work products and tasks involved in the software process. In particular, it identifies the variable process elements whose inclusion is defined by the rules [27]. The second input is the *organizational projects context model* (component 2 in Fig. 1), where the attributes that characterize projects along with their set of potential values are defined. The graphical environment itself (component 3 in Fig. 1) includes two elements: an interactive rule definition interface for defining transformation rules and the Text-to-Model transformation for generating a VDM.
– *The variation decision model (VDM)* for formally representing transformation rules. We have defined the VDM using our DSL (component 4 in Fig. 1) that is a high-level representation of the transformation rules using an abstract syntax (metamodel) and a concrete syntax (textual representation). The VDM is inspired by Decision Models [32] and Semantics of Business Vocabulary and Business Rules (SMVB) used for building decision rules [19]. In this work, we also defined a Variation Decision Metamodel (VDMM).
– *The tailoring transformation generation (HOT)* (component 5 in Fig. 1). This HOT has one input model, the VDM, and follows a transformation synthesis pattern for generating the tailoring transformation model, that

conforms to the ATL metamodel. This HOT is an exogenous transformation because the models are expressed using different languages (decision and transformation models). Then, we apply an ATL extractor for generating the tailoring transformation. In this work, we applied a vertical transformation because the source and target model of the HOT reside at different abstraction levels.

– *The tailoring transformation.* After applying the HOT, we obtain a tailoring transformation (component 6 in Fig. 1) using an ATL subset. The DSTL uses only some structures of the ATL language that are necessary in this domain (e.g., lazy rules and called rules are not used). The tailoring transformation is generated only once and it can be used for tailoring all concrete contexts. The *adapted software process model* (component 8 in Fig. 1) is the output of applying such transformation using the organizational software process and the concrete project context as input. Such a process includes the process elements that are needed for the specified context.

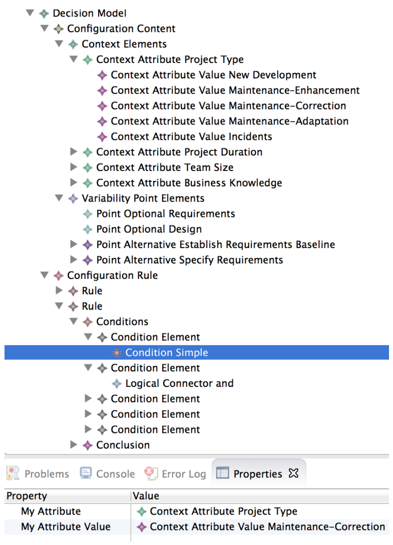
This research has two specific goals: (1) automatically generating tailoring transformations using our DSL, and (2) applying the generated tailoring transformation for obtaining an adapted software processes. The results of this work will be applied in organizational software process of Chilean SSE, so that they can be adapted to different project contexts, and thus the development productivity and product quality are improved. The expressiveness of the DSTL and the usability of the graphical environment will be rigorously evaluated.

5 Preliminary Results and Contributions

We have studied several alternatives for implementing the preliminary solution of the proposal, and we have analyzed the feasibility and the effort required to implement the ATR components.

The first step was to explore the feasibility to implement HOTs using two input models. We have successfully addressed this issue, in our case the organizational software process model and the organizational context model. We tried using some of the most promising techniques and tools for developing HOTs such as AMW and MOFScript but we found limitations for dealing with two input models [25]. We then decided to divide the HOT in two parts: rule definition and transformation generation. We defined one input model for the HOT and we called it VDM. We found researches about business rules [19] and decision models [32], we used them as inspiration and we formalized the DSL for our application. The VDM is a formal representation of transformation rules.

The second step consisted of building a proof of concept and an initial implementation of the ATR. For this step, we were able to build the infrastructure of the ATR and apply it through a running example. The initial ATR implementation has a graphical interface, a preliminary VDM (Fig. 2), a HOT and an initial DSTL. For the first implementation of ATR, we defined the models using Eclipse Modeling Tools (EMT) and implemented the transformations using Java. The running example was used to generate a tailoring transformation for a Chilean



```
▼ Decision Model
  ▼ Configuration Content
    ▼ Context Elements
      ▼ Context Attribute Project Type
          Context Attribute Value New Development
          Context Attribute Value Maintenance-Enhancement
          Context Attribute Value Maintenance-Correction
          Context Attribute Value Maintenance-Adaptation
          Context Attribute Value Incidents
      ▶ Context Attribute Project Duration
      ▶ Context Attribute Team Size
      ▶ Context Attribute Business Knowledge
    ▼ Variability Point Elements
        Point Optional Requirements
        Point Optional Design
      ▶ Point Alternative Establish Requirements Baseline
      ▶ Point Alternative Specify Requirements
  ▼ Configuration Rule
    ▶ Rule
    ▼ Rule
      ▼ Conditions
        ▼ Condition Element
            Condition Simple
        ▼ Condition Element
            Logical Connector and
        ▶ Condition Element
        ▶ Condition Element
        ▶ Condition Element
      ▶ Conclusion
```

Property	Value
My Attribute	Context Attribute Project Type
My Attribute Value	Context Attribute Value Maintenance-Correction

Fig. 2. The transformation rules definition using our DSL

SSE [26]. The output of the Model-Based tool is an ATL file that is compilable in EMT for software process tailoring.

The third step encapsulated ATR in a megamodel application. We have proposed a megamodel to improve the adoption and evolution of the MDE-based tailoring approach in industry [3]. ATR is a relevant part of this megamodel.

6 Plan for Evaluation and Validation

We have applied ATR to an exploratory case study in a Chilean company, and we will apply a confirmatory multi-case study in two or three other companies.

The exploratory case study addressed the evaluation of correctness and productivity. We compared the productivity of adapted software processes obtained by using two different approaches: template-based tailoring [9] and automatic tailoring [26]. The productivity is measured in terms of the missing and extra tasks the template-based tailored process has when compared with the automatically tailored process. We found that the automatic tailoring has more fine-grained rules for generating the process and the results of template-based tailoring are in most of the cases sub-optimal [12]. The correctness was evaluated by comparing the adapted software processes obtained manually and automatically. We tested our automatic tailoring by applying it to the company's process, and then interviewing the process engineer about the quality of the result.

In the multi-case study, we will incorporate the evaluation of usability and expressiveness. We will evaluate the usability by applying structured questionnaires to process engineers [8] about their perception about the graphical interface usability focusing on the elements of Quality in Use Integrated Measurement

(QUIM) [22]. Evaluating the expressiveness of the DSTL will consist of verifying that it counts on all the required constructs for expressing the process engineer's intentions for tailoring software process. Usability and expressiveness will be analyzed using qualitative methods [24], while correctness and productivity will use quantitative methods [11].

We expect that SSEs will improve their competitiveness using the proposed approach because they will be able to apply software processes that are specifically adjusted to the needs of their particular projects. Using the solution will probably also enhance the quality of the software products, although this evaluation is out of the scope of this work.

Acknowledgments. This work has been partly funded by Project Fondef GEMS IT13I20010 and Luis Silvestre was also supported by PhD Scholarship Program of Conicyt, Chile (CONICYT-PCHA/2013-63130130).

References

1. D. Balasubramanian, A. Narayanan, C. vanBuskirk, and G. Karsai. The graph rewriting and transformation language: GReAT. *Electronic Communications of the EASST*, 1, 2007.
2. K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema. Developing applications using model-driven design environments. *Computer*, 39(2):33–40, 2006.
3. M. C. Bastarrica, J. Simmonds, and L. Silvestre. Using megamodeling to improve industrial adoption of complex MDE solutions. In J. M. Atlee, V. Kulkarni, T. Clark, R. B. France, and B. Rumpe, editors, *MiSE*, pages 31–36. ACM, 2014.
4. J. Bézivin. Model driven engineering: an emerging technical space. In H. Berlin, editor, *GTTSE*, pages 36–64, Braga, Portugal, 2006. Springer-Verlag.
5. J. Bézivin, F. Büttner, M. Gogolla, F. Jouault, I. Kurtev, and A. Lindow. Model transformations? transformation models! In O. Nierstrasz, J. Whittle, and D. Harel, editors, *MoDELS*, volume 4199 of *LNCS*, pages 440–453. Springer, 2006.
6. J. Bézivin, G. Dupé, F. Jouault, G. Pitette, and J. E. Rougui. First experiments with the ATL model transformation language: Transforming XSLT into XQuery. In *OOPSLA*, page 50, Anaheim, CA, USA, 2003.
7. E. Breton and J. Bézivin. Model driven process engineering. In *COMPSAC*, pages 225–230. IEEE, 2001.
8. M. A. Campion, J. E. Campion, and J. P. Hudson. Structured Interviewing: A note on Incremental Validity and Alternative Questions Types. *Journal of Applied Psychology*, 79(6):998–1002, 1994.
9. A. Cockburn. *Crystal Clear a Human-powered Methodology for Small Teams.* Addison-Wesley Professional, first edition, 2004.
10. M. D. Del Fabro, J. Bézivin, and P. Valduriez. Weaving Models with the Eclipse AMW plugin. In *Eclipse Modeling Symposium*, volume 2006. Citeseer, 2006.
11. T. Dybå, V. B. Kampenes, and D. I. K. Sjøberg. A systematic review of statistical power in software engineering experiments. *Information & Software Technology*, 48(8):745–755, 2006.
12. F. Gonzalez, L. Silvestre, M. Solari, and M. C. Bastarrica. Template-Based vs. Automatic Process Tailoring. In *To appear in proc. SCCC*, 2014.

13. J. Gray, Y. Lin, and J. Zhang. Automating change evolution in model-driven engineering. *Computer*, 39(2):51–58, 2006.
14. J. A. Hurtado, M. C. Bastarrica, A. Quispe, and S. F. Ochoa. An MDE approach to software process tailoring. In D. Raffo, D. Pfahl, and L. Zhang, editors, *ICSSP*, pages 43–52, Honolulu, HI, USA, 2011. ACM.
15. J. A. Hurtado, M. C. Bastarrica, A. Quispe, and S. F. Ochoa. MDE-based process tailoring strategy. *Journal of Software: Evolution and Process*, 26(4):386–403, 2014.
16. J. Hutchinson, J. Whittle, and M. Rouncefield. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, 89:144–161, 2014.
17. A. Kalnins, J. Barzdins, and E. Celms. Model Transformation Language MOLA. In U. Aßmann, M. Aksit, and A. Rensink, editors, *MDAFA*, volume 3599 of *LNCS*, pages 62–76, Twente, The Netherlands, 2004. Springer.
18. M. Kuhrmann. You can't tailor what you haven't modeled. In H. Zhang, L. Huang, and I. Richardson, editors, *ICSSP*, pages 189–190. ACM, 2014.
19. OMG. Semantics of Business Vocabulary and Business Rules (SBVR) Version 1.0. Technical Report 2008-04-01, 2008. http://www.omg.org/spec/SBVR/1.0/.
20. OMG. Software & Systems Process Engineering Metamodel Specification (SPEM). Technical Report 2008-04-01, 2008. http://www.omg.org/spec/SPEM/2.0/PDF/.
21. OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Technical report, January 2011. http://www.omg.org/spec/QVT/1.1/PDF/.
22. H. Padda. *QUIM: A Model for Usability/Quality in Use Measurement.* LAP Lambert Academic Publishing, Germany, 2009.
23. D. C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.
24. C. B. Seaman. Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Trans. Software Eng.*, 25(4):557–572, 1999.
25. L. Silvestre, M. C. Bastarrica, and S. F. Ochoa. HOTs for Generating Transformations with Two Input Models. In *proc. SCCC*, 2013.
26. L. Silvestre, M. C. Bastarrica, and S. F. Ochoa. A model-based tool for generating software process model tailoring transformations. In L. F. Pires, S. Hammoudi, and J. Filipe, editors, *MODELSWARD*, pages 533–540. SciTePress, 2014.
27. J. Simmonds, M. Bastarrica, L. Silvestre, and A. Quispe. Variability in software process models: Requirements for adoption in industrial settings. In *PLEASE*, pages 33–36, May 2013.
28. Y. Sun, J. White, and J. Gray. Model Transformation by Demonstration. In A. Schürr and B. Selic, editors, *MoDELS*, volume 5795 of *LNCS*, pages 712–726, Denver, CO, USA,, 2009. Springer.
29. M. Tisi, J. Cabot, and F. Jouault. Improving Higher-Order Transformations Support in ATL. In *ICMT*, volume 6142 of *LNCS*, pages 215–229, Malaga, Spain, 2010. Springer.
30. M. Tisi, F. Jouault, P. Fraternali, S. Ceri, and J. Bézivin. On the Use of Higher-Order Model Transformations. In *ECMDA-FA*, volume 5562 of *LNCS*, pages 18–33, Enschede, The Netherlands, 2009. Springer.
31. D. Varró and A. Balogh. The model transformation language of the VIATRA2 framework. *Science of Computer Programming*, 68(3):214–234, 2007.
32. D. Weiss, J. Li, H. Slye, T. Dinh-Trong, and S. Hongyu. Decision-Model-Based Code Generation for SPLE. In *SPLC*, pages 129–138, Sept 2008.
33. M. Wimmer, M. Strommer, H. Kargl, and G. Kramler. Towards Model Transformation Generation By-Example. In *HICSS*, page 285, Big Island, HI, USA, 2007. IEEE Computer.