A Usable MDE-based Tool for Software Process Tailoring

Luis Silvestre, María Cecilia Bastarrica and Sergio F. Ochoa
Computer Science Department, University of Chile
Beauchef 851, Santiago - Chile
{lsilvest,cecilia,sochoa}@dcc.uchile.cl

***Abstract*—In order to systematize development, software companies define their organizational processes. The process engineer is in charge of this activity. Tailoring software processes is an activity that allows project managers to adapt organizational software processes to the needs of particular projects. Model-driven engineering (MDE) has been applied with that purpose using process model tailoring transformations. Although this approach is technically feasible, there are still factors that jeopardize its usage in industry. First, current transformation languages and tools are not simple for defining and applying tailoring transformations. Second, the potential users -process engineers and project managers- do not usually have the required knowledge for writing transformations. Trying to deal with these usability challenges, this paper proposes an MDE-based tool that allows defining the models for the organizational software process and the organizational context, as well as tailoring rules using a usable user interface, so that the project manager just requires to define the project context of the particular project in order to automatically obtain the tailored process. This usable interface hides the complexities of the tool backend built as a megamodel, requiring from the process engineer and the project manager only knowledge about project characteristics and how they affect tailoring. We report our experience of applying the tool in a real-world software process and we outline this experience in http://www.dcc.uchile.cl/gems/docs/DemoMODELS2015.pdf.**

I. Introduction

Software process formal specification allows companies to rigorously document their development process, and tool support for process analysis and evolution. Development projects faced by a particular company may be of different kinds, e.g., large or small, complex or simple, new development or evolution, and therefore the same process is not equally appropriate for all of them. Process tailoring is the activity of adapting a general process to match the needs of the project at hand. Such a tailoring activity is generally performed by the process engineer and the project manager: the process engineer knows about the organizational process and how project characteristics affect tailoring, while the project manager knows about the project's particular characteristics. There are several proposals for process tailoring, such as using the same process for addressing all projects, counting on a family of predefined processes or configuring a process by putting together appropriate pieces [7]. We have worked with MDE-based process tailoring for the last five years and it has proved to be technically feasible [3].

MDE-based tailoring consists of defining an organizational software process model along with its variability and the project context model, and then use these models as input of a tailoring transformation, whose output is the project adapted process model [5]. Writing tailoring transformations requires knowledge about model transformation programming languages, and also about the software process model and the way its variability should be resolved according to the project context. These two kinds on knowledge –programming a model transformation and configuring a software process– are almost never mastered by the same person in the company, and it is even less frequent in small software companies, where the average expertise of the staff is not high [8]. Moreover, even if there is someone that counts on both kinds of knowledge, manually writing complex model transformations is still an error-prone activity.

As part of a previous work [10], we have built a prototype toolset that allows specifying tailoring rules through a user interface, and automatically generating the corresponding tailoring transformations. This toolset was composed of a set of loosely coupled tools that allowed defining just simple optional rules. However, while testing it in industrial scenarios, we have realized that more complex tailoring rules are required; alternative options should be chosen as part of the tailoring activity, and there are some rules that require compound conditions. Moreover, the toolset was not usable by our target users since it needed dealing with intermediate models and transformations, not for building them but to run each tool with the appropriate models and transformations.

In this paper we present an integrated MDE-based tool that allows the process engineer defining the organizational process and context as well as the transformation rules, and provides the project manager an interface for defining the project context and obtaining the tailored software process. All these activities are executed without the need to manipulate models or transformations. The tool's complete structure is shown in Fig. 1. Notice that its back-end is built as a megamodel [11] and the front-end contains all the user interfaces.

In this paper we show the application of the tool in the case of one of our industrial partners, Amisoft, a small Chilean software company. The case study includes the activities of the process engineer –the software process, context and rule definition–, the transformation generation and the project manager activities –the project context definition and the tailoring transformation execution–. We also show how this improved tool is able to generate transformations that produce the correct

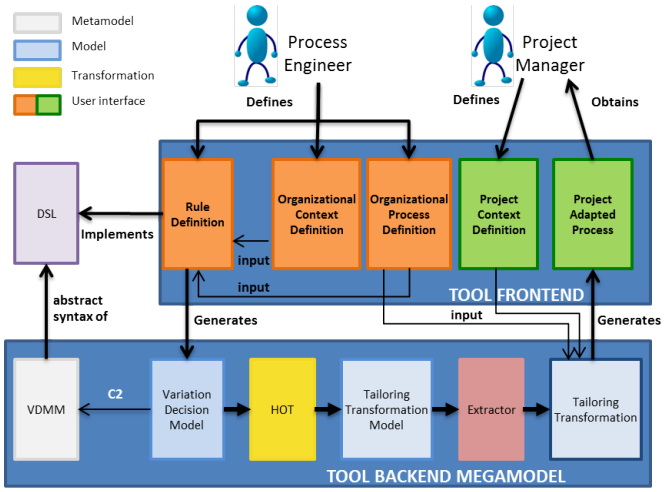

Fig. 1. MDE-based tool general structure

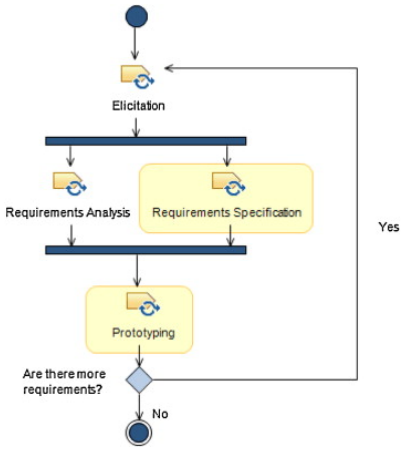

Fig. 2. Amisoft's *Requirement* activity

tailored software processes in a usable manner, only requiring from the process engineer and the project manager knowledge about the software process tailoring, but hiding the complexity of managing models and writing model transformations. This integrated tool is also more expressive than its previous version, being able to define more sophisticated rules.

Section II presents related work concerning MDE-based tailoring and transformation generation. Section III describes how the process engineer interacts with the tool, Sec. IV how the tailoring transformation is generated, and Sec. V the interaction of the project manager with the tool. Finally, Sec. VI presents the conclusions and describes ongoing work.

II. Related Work

Proposals trying to address MDE solutions should balance the formality required by MDE and the usability needed by the process engineers. MOLA [4] and GREaT [1] allow specifying transformation rules through visual mapping patterns and VIATRA [15] provides a textual rule editor for specifying patterns. These proposals still need the process engineer to interact with metamodels, models, classes or code to adjust them to match the features of a specific project. Therefore, if the user does not count on the required experience for managing these elements, he could neither write nor maintain the transformation code.

Sijtema [9] proposes an extension to the Atlas Transformation Language (ATL), which is based on feature models and requires the specification of the so-called variability rules. In the approach, variability rules are transformed to standard ATL code, i.e., variability is translated to *called rules* in ATL using a HOT. Although the proposal raises the abstraction level, the process engineer still needs to understand the ATL extension and interact with code.

AToMPM [13] is a Web-based metamodeling and transformation tool for multi-paradigm modeling. It allows defining DSLs through an interactive interface for specifying their abstract syntax. It also provides support for rule definition and scheduling using graph transformation rules. In this sense their approach is similar to ours. However, in AToMPM rules still need to be specified using a custom language, and it only allows for simple conditions.

There are also proposals such as MTBE (Model Transformations By Example) [14] and MTBD (Model Transformation By Demonstration) [12], which present solutions for simplifying the implementation of model transformations by using strategies and patterns with a visual support. They both generate part of the transformation code, but the user still needs to complete the generated code, and thus the transformation rule generation becomes a semi-automatic process. Both proposals highlight the need for developing new solutions that simplify the rule definition, so that they become usable solutions for non-experts in MDE.

III. Process Engineer's User Interface

The process engineer is in charge of defining the organizational process model, the organizational context model, and the variation decision model, i.e., the model that represents the tailoring rules. Our tool integrates user interfaces for performing each of these activities. We show the application of the tool for the case of Amisoft, one of our industrial partners.

Amisoft is a small Chilean software company. For the last 6 years, it has defined and enacted its software process.We have applied the previous MDE-based software process tailoring toolset in Amisoft [2] and, although they validated the correctness of the resulting tailored processes, they admitted that building tailoring transformations was beyond their capabilities, even though they counted on a vast experience with formalized software processes. In the following sections we show how the new integrated tool improved this situation.

A. Organizational Process Definition

Several Chilean software companies are already using EPF Composer as the framework for defining their software process, even though there are other available tools [11]. We will include this tool as part of our solution in order to enhance adoption. It allows defining the process breakdown, including

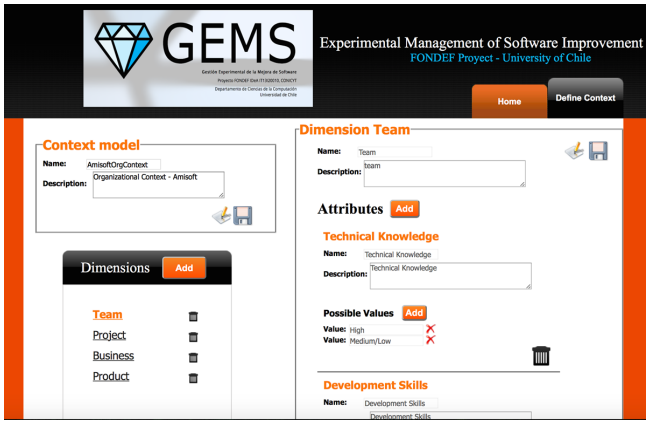

Fig. 3. Amisoft's Organizational Context Definition

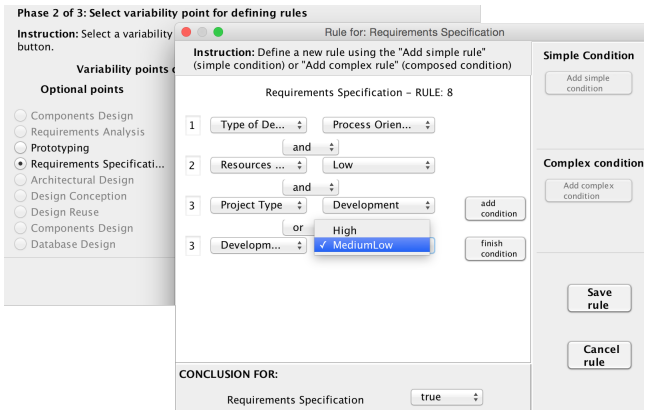

Fig. 4. User interface for interactive tool for rule definition

primitives for specifying process element variability, as well as process behavior using activity diagrams. EPF Composer allows exporting the software process as an XML file that is used as input for other activities. Figure 2 shows the *Requirements* activity, that is a portion of Amisoft's software process. Notice that *Requirement Specification* and *Prototyping* are highlighted in the diagram for indicating optionality, even though this is not standard SPEM [6]; internally in the tool, they are formally specified as optional.

B. Organizational Context Definition

We have developed a user interface for defining the organizational context. Here, the process engineer defines the attributes that may determine how the process should be tailored, as well as their potential values. For example, in Fig. 3, we see that the attribute *Technical Knowledge* may take values *High* or *Medium/Low*. The user interface allows editing this organizational context by adding or deleting attributes, and/or modifying their potential values. The output of this interface will be saved as the *Organizational Context Model.*

C. Tailoring Rules Definition

By keeping in mind the goal of reducing the skills required to define tailoring transformation rules, we developed a tool

```
-- Optional Points
helper def: optionalRule(name:String): Boolean =
                    if('Prototyping'= name) then
                        thisModule.ruleOpt9()
                    else (
                    if('Requirements Specification'= name) then
                        thisModule.ruleOpt8()
```

Fig. 5. Generated Amisoft Optional Points

that allows the process engineer to create models through a graphical user interface. This interface allows defining the tailoring rules for each variation point of the software process according to the values in the context; therefore, the interface takes the *Organizational Software Process* and the *Organizational Context Model* previously defined as input as shown in Fig. 4. This tool implements a DSL formally defined in [10].

This activity involves two steps, the definition of tailoring rules and the automatic generation of a Variation Decision Model (VDM). In the first step, the tool guides the users presenting them only the process elements defined as variable and allowing them specify a rule either simple or complex for each variation point. The tool also counts on a menu where the context attributes and their potential values can be selected not requiring any typing. The second step is done without the user intervention.

We define process tailoring rules as a set of conditions, conclusions and logical operators that decide an action about a variable process element. Consequently, a VDM is a set of rules of the form $condition \Rightarrow conclusion$, where a $condition$ is a predicate about the application domain, and the $conclusion$ indicates how a variation point is resolved when the condition holds. A $condition$ may be simple, i.e., when it states $attribute = value$, or it may be complex when it is formed by a series of simple conditions combined with logical operations. The definition of the tailoring rule associated to the *Requirements Specification* task is shown in Fig. 4.

IV. Tailoring Transformation Generation

Transformation generation has two steps: generate the tailoring transformation model and extract the tailoring transformation code, as shown as part of the megamodel back-end in Fig. 1. This activity is completely automatic and transparent for both users. It is only generated once and can be used to tailor all particular project contexts.

We have built a Higher Order Transformation (HOT) that takes the VDM as input, and uses a transformation synthesis pattern for generating the tailoring transformation model using a M2M transformation. This HOT is an exogenous and vertical. The resulting tailoring transformation model conforms to the ATL metamodel. The tailoring transformation code is obtained through a M2T transformation that takes the tailoring transformation model as input.

Figure 5 shows an excerpt of the generated tailoring transformation where variable process elements are identified as well as their corresponding helper rules for resolving their inclusion in the tailored process. Figure 6 shows the rules for *Prototyping* and *Requirements Specification.* We can see that

```
helper def:ruleOpt8():Boolean=
    if(thisModule.getValue('Type of Deliverable') = 'Product Oriented'
        or thisModule.getValue('Resources Availability') = 'Low'
        or (thisModule.getValue('Project Type') = 'Maintenance'
            and thisModule.getValue('Development Skills') = 'High'))
        then false
    else true
endif;

helper def:ruleOpt9():Boolean=
    if(thisModule.getValue('Project Type') = 'Maintenance')
    then false
    else true
endif;
```

Fig. 6. Generated Amisoft Tailoring Transformation

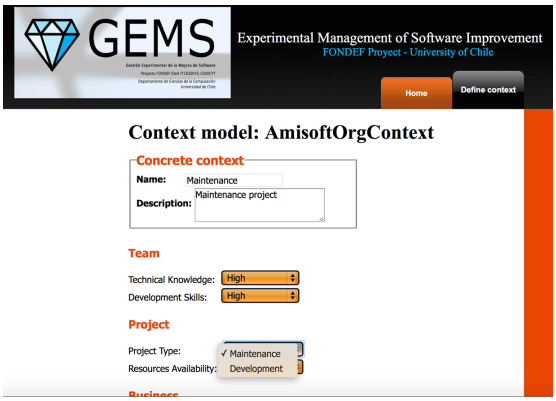

Fig. 7. Amisoft's Project Context Definition

rule *condition* for *Requirements Specification* is the same as that specified with the tool in Fig. 4.

V. Project Manager's User Interface

The project manager is the final user of the tailoring tool since he is in charge of executing the tailored process in each project. To this end, he defines the context of the project at hand, and the tool returns the tailored software process.

We have built a tool for defining the project context. Figure 7 shows the use of this tool for defining a *Maintenance* project where *Technical Knowledge* is *High* and *Development Skills* is *High* too. With these conditions, and according to rules included in Fig. 6, neither *Prototyping* nor *Requirements Specification* will be included in the tailored software process since it is a *Maintenance* project. However, if it were a *Development* project, *Prototyping* would be included, but *Requirements Specification* would still not be included since *Development Skills* is *High*.

VI. Conclusions and Future Work

This article presents an MDE-based integrated tool for defining and executing software process tailoring. It provides a usable user interface that allows its users -the process engineer and the project manager- to deal only with process and context-related concepts hiding all the models and transformations involved. The tool is better than its predecesor [10] in two ways: it is an integrated tool that hides even the existence of models and transformations and not only their generation, and it is more expressive since it allows specifying more complex tailoring rules. These enhancements were motivated by the application of the tool in three of our industrial partners.

We are currently applying the tool in two other small Chilean companies. So far, the results are promising. Nevertheless, we are aware that the reality of different software companies may be also different, and thus the applicability of our tool is not guaranteed. For instance, it may be the case that some conditions for resolving variability do not just depend on context values, but also on the way other variation points are resolved; the tool does not support this kind of conditions yet but we have not found the need in practice.

Acknowledgements

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] Daniel Balasubramanian, Anantha Narayanan, Chris vanBuskirk, and Gabor Karsai. The graph rewriting and transformation language: GReAT. *Electronic Communications of the EASST*, 1, 2007.

[2] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, Sergio F Ochoa, and Jocelyn Simmonds. MDE software process lines in small companies. *Journal of Systems and Software*, 2012.

[3] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, Alcides Quispe, and Sergio F. Ochoa. MDE-based process tailoring strategy. *Journal of Software: Evolution and Process*, 26(4):386–403, 2014.

[4] Audris Kalnins, Janis Barzdins, and Edgars Celms. Model Transformation Language MOLA. In *In proceedings of MDAFA 2004*, volume 3599 of *LNCS*, pages 62–76, Twente, The Netherlands, 2005. Springer.

[5] Georg Kalus and Marco Kuhrmann. Criteria for Software Process Tailoring: A Systematic Review. In *In proceedings of ICSSP 2013*, pages 171–180, New York, NY, USA, 2013. ACM.

[6] OMG. Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0. Technical Report 2008-04-01, Object Management Group, 2008. http://www.omg.org/spec/SPEM/2.0/PDF/.

[7] Oscar Pedreira, Mario Piattini, Miguel R. Luaces, and Nieves R. Brisaboa. A Systematic Review of Software Process Tailoring. *SIGSOFT Softw. Eng. Notes*, 32(3):1–6, 2007.

[8] Alcides Quispe, Maira Marques, Luis Silvestre, Sergio F. Ochoa, and Romain Robbes. Requirements engineering practices in very small software enterprises: A diagnostic study. In *In Proceedings of SCCC 2010, Antofagasta, Chile*, pages 81–87. IEEE Computer Society, 2010.

[9] Marten Sijtema. Introducing Variability Rules in ATL for Managing Variability in MDE-based Product Lines. In Marcos Didonet Del Fabro, Frédric Jouault, and Ivan Kurtev, editors, *Proceedings of the 2nd International Workshop on Model Transformation with ATL*, volume 10, pages 39–49, Malaga, Spain, 2010. CEUR Workshop.

[10] Luis Silvestre, María Cecilia Bastarrica, and Sergio F. Ochoa. A Model-based Tool for Generating Software Process Model Tailoring Transformations. In *In proceedings of MODELSWARD 2014, Lisbon, Portugal*, pages 533–540. SciTePress, 2014.

[11] Jocelyn Simmonds, Daniel Perovich, and María Cecilia Bastarrica. A Megamodel for Software Process Line Modeling and Evolution. In *To appear in Proceedings of MODELS 2015*, Ottawa, Canada, 2015.

[12] Yu Sun, Jules White, and Jeff Gray. Model Transformation by Demonstration. In *In proceedings of MoDELS 2009*, volume 5795 of *LNCS*, pages 712–726, Denver, CO, USA,, 2009. Springer.

[13] Eugene Syriani, Hans Vangheluwe, Raphael Mannadiar, Conner Hansen, Simon Van Mierlo, and Hüseyin Ergin. Atompm: A web-based modeling environment. In *Demos/Posters/StudentResearch@ MoDELS 2013*, pages 21–25. CEUR Workshop Proc., 2013.

[14] Dniel Varr. Model transformation by example. In *In proceedings of MoDELS 2006*, volume 4199 of *LNCS*, pages 410–424. Springer Berlin Heidelberg, 2006.

[15] Dániel Varró, Gergely Varró, and András Pataricza. Designing the automatic transformation of visual languages. *Science of Computer Programming*, 44(2):205–227, 2002.