An MDE Approach to Software Process Tailoring

Julio A. Hurtado Alegría
Computer Science Dept.
Universidad de Chile
IDIS Research Group
University of Cauca
jhurtado@dcc.uchile.cl

María Cecilia Bastarrica
Computer Science Dept.
Universidad de Chile
cecilia@dcc.uchile.cl

Alcides Quispe
Computer Science Dept.
Universidad de Chile
aquispe@dcc.uchile.cl

Sergio F. Ochoa
Computer Science Dept.
Universidad de Chile
sochoa@dcc.uchile.cl

ABSTRACT

Defining organizational processes is essential for enhancing maturity. However the best process depends on the particularities of each project. Typically a process engineer defines a specific process for each project in an ad-hoc fashion, which is expensive, unrepeatable and error prone. Trying to deal with this challenge we propose a model-based approach to software process tailoring that generates project specific processes based on the organizational process and the project context. The approach is systematic, repeatable and it does not depend on the people using it. The proposal has been applied for tailoring the Requirements Engineering process of a medium size company. The obtained results were validated by process engineers of the company. Processes obtained using the proposed approach matched the ones used in the company for planned contexts and also they were reasonable for non-expected situations.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*software process models*

General Terms

Management, Reliability

Keywords

Software process lines, tailoring, model-driven engineering

1. INTRODUCTION

Different software development life cycles suggest specific activities to be carried out in a particular order, from traditional models such as the Waterfall, to more modern ones such as RUP, Scrum or XP. Moreover, if a company aims to certify or evaluate its software development process, it should be rigorously defined as prescribed by most popular models and standards as CMMI-Dev and ISO/IEC 12207. This organizational process definition always requires an enormous effort and it still needs to be adapted to satisfy the specific characteristics of different project situations [22].

There is no unique software process since appropriateness depends on various organizational, project and product characteristics, and what is even worse, all these characteristics evolve continuously. A one-size fits-all approach does not work for software development [14]. Each project has its own characteristics and requires a particular range of techniques and strategies [21], and selecting a set of practices and integrating them into a coherent process should also be aligned with the business context [9]. In their process improvement approach, Dorr et al. suggest that the right set of practices for a project can be better found if we understand the context of the company [13]. Based on these findings we follow the idea that each project context should dictate the definition of the process that best fits it. Moreover, the particular process applied should not vary dramatically from one project to the other, so that process knowledge acquired by the development team could be reused.

Tailoring is the process through which a general software process is configured for adapting it to a project's particularities [28]. Empirical studies show that process tailoring is difficult because it involves intensive knowledge generation and deployment [30] and it is also time consuming [24]. Moreover, the knowledge necessary for a good tailoring may be lost from one project to the following one. Therefore, the tailoring process is unrepeatable and difficult to evolve.

Model-driven engineering (MDE) [31] is a software development approach in which abstract models are defined and systematically transformed into more concrete models, and eventually into source code. This approach promotes reuse through a generative strategy. MDE can also be used in software process engineering [7]: using transformations as instantiation strategies [19].

In this paper we propose an approach for automatically tailoring organizational processes to particular project contexts, based on MDE techniques so that appropriate processes are achieved rapidly and with little effort. Tailoring is implemented by means of a model transformation whose inputs are the organizational process model including vari-

ICSSP '11, May 21-22, 2011, Waikiki, Honolulu, HI, USA

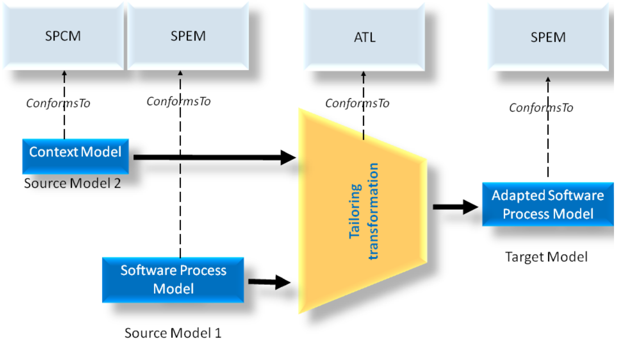


Figure 1: A generative strategy for process tailoring

abilities and a model of the project context, and whose output is the context-adapted process, as shown in Fig. 1. We formalize metamodels and implement transformation rules using ATL (Atlas Transformation Language).

Using the proposed approach, this paper formalizes the requirements engineering (RE) process that has been used and evolved for several years in a medium size software company. The process considers variation points according to different context attributes including the knowledge about the application domain (high, medium or low), the project type (development, extension or reengineering) and size (small, medium or large), among others. Combining the values these attributes may take, we would need different particular processes. The paper formalizes the general RE process including its planned variability, and it shows how a model transformation is actually able to yield the particular process to be followed in each specific context. We were also able to achieve appropriate processes by combining tailoring rules for unanticipated settings.

The rest of the paper is structured as follows. Section 2 presents some related work. In Sect. 3 we describe the tailoring process and the involved models and transformations. The application of the tailoring technique for the case of RE is presented in Sect. 4. Finally, Sect. 5 presents the conclusions and future work.

2. RELATED WORK

There are several diverse approaches to tailoring processes. The *assamble approach* [12] enables the implementation of tailoring decisions about deleting and merging process elements. These works use formalisms that turn process tailoring a very complex task in practice.

The *situational method engineering* (SME) aproach focuses on project specific method construction [29]. During the organizational process definition, an adaptable structure and a guide for process tailoring by situational knowledge is defined [1]. Nevertheless, in most cases the effort for tailoring the process is huge, especially when an assembly approach is carried out at tailoring time [22]. This is a big problem because process tailoring normally is the responsibility of the project manager, but requires the experience and knlowledge of the software process engineer, so a suitable separation beetween their roles is not achieved [4].

Some processes as the Unified Process use an *adjustment guide approach* where tailoring rules are defined as recommendations to adapt phases, iterations and disciplines according to project specific situations. This was the approach originally followed in the company where we validated our proposed MDE tailoring approach.

Agile methods such as XP use an *auto-adaptable approach* where a project and team-adapted process results as an emergent entity from a set of principles, values and practices. Other processes as Crystal Methodology follow a *template based approach*, where a methodology family with four members: Clear, Yellow, Orange and Red is defined. Commercial processes such as Rational Unified Process, use a *framework based approach* or configuration approach [5], where a general process is defined and a specific configuration is created for each specific project. The framework strategy makes the process model large and complex, and process engineering knowledge is required to produce a valid configuration, whereas in the template strategy it is difficult to define the adequate set of templates for satisfying each specific project [8].

A *recovery tailoring approach* has been proposed using case based reasoning [15, 36] and neural networks [27]. In these cases, tailoring is based on an incremental set of previously tailored processes, so the benefits are achieved after various processes have been adapted. The main difficulties in this approach is the set up cost required and the non planned change and evolution of various processes, instead of just one.

Killisperger [19] proposes an *instantiation based approach*. Because the industry has few processes formalized up to the enactment level, this approach may result in little benefit in practice.

Provided that software processes can be considered as software too [26], a Software Process Line (SPrL) can be considered as a special Software Product Line (SPL) in the software process engineering domain. SPrL share common features and exhibit variability [33]. Consequently, a SPrL is an ideal way to define, tailor and evolve a set of related processes as it is established by the works on process varibility representation [32], SPrL architectures [35], process domain analysis [24], and SPrL scoping [3]. A *SPrL approach* facilitates planned reuse, while classic tailoring re-actively integrates unanticipated variability in the process model [3].

Our work proposes a *MDE tailoring strategy* as a production strategy of project-specific process models in the context of a SPrL. We use the MDE tailoring strategy using as input an organizational process model with variabilities and a specific context model. The context of the software process has gained importance, but it has been usually represented informally. Armbrust et al. [2] define three dimensions to define the characteristics in the SPrL scope definition: product, project and process. The COCOMO II model [6] defines a set of attributes and dimensions to estimate a project, that are useful for representing context models too. The Incremental Commitment Model Process [20] defines a set of patterns for rapid-fielding using contextualized information. However, these contexts are specific to a process, organization or research issue. In order to help organizations determining their relevant dimensions and context attributes we have defined a Software Process Context Metamodel following the initial ideas presented in [16]. The process model variability is represented using a process feature model similar to software features [11] and implemented as a SPEM 2.0 process. So, the MDE Strategy helps achieving a separation between the process modeling stakeholders and process

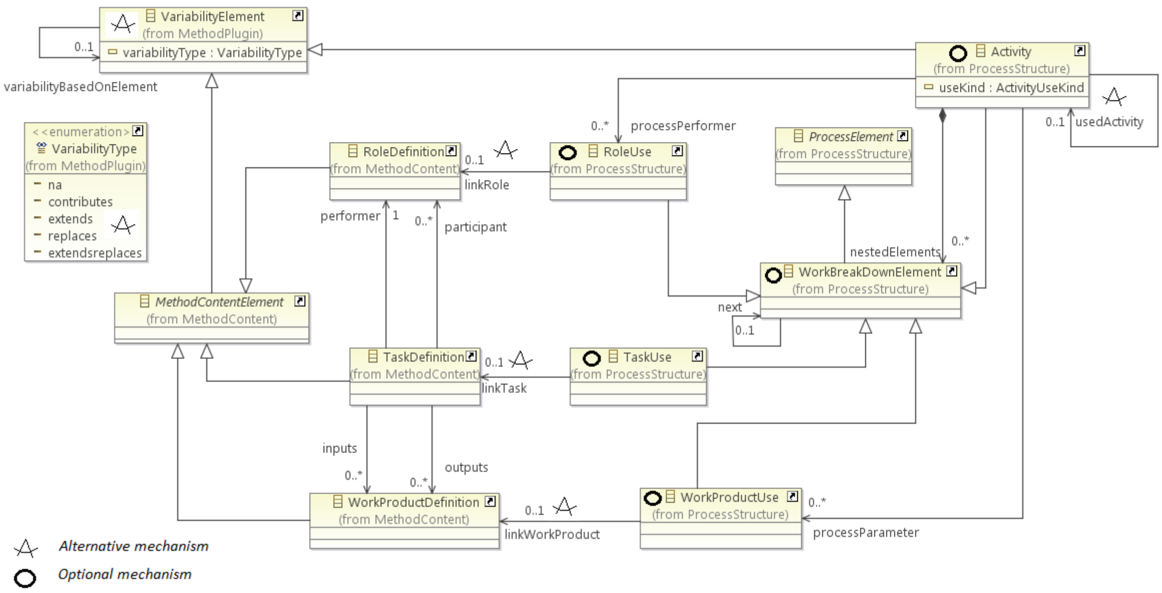


Figure 2: Experimental SPEM (eSPEM) highlighting where variability is specified

enactment (project) stakeholders [4] and hides the complexity by intensively reusing tailoring knowledge. Furthermore, the *MDE tailoring strategy* provides a way to cost-efficiently instantiate a general process model into project-specific process models where the project manager should only provide a definition of a specific situation.

3. TAILORING THE SOFTWARE PROCESS

Defining an organizational software process is necessary if a company wants to improve its development process, and completely required in order to achieve an evaluation or certification such as CMMI or ISO/IEC 12207. Although defining and documenting the process demands an important effort, a general process is still not appropriate for all projects, even within the same organization. Moreover, an organization that usually develops certain type of projects using a particular process, may eventually get engaged in a different type of project, and thus the processes that have always worked fine become inadequate. Defining a customized process for each project is too expensive due to the amount of resources from the project itself it would consume. Having a set of predefined processes for a series of different contexts implies a high maintenance cost, and still does not assure to cover all possible contexts. Therefore, tailoring the organizational process seems to be a good trade-off.

We first define the models and metamodels involved in the proposed tailoring approach, and then the ATL transformations that implement the tailoring process are presented. Finally a brief description of the implemented tool is included.

3.1 Models

The organizational process is defined as a SPEM 2.0 process model including variabilities. A context model is defined to express each one of the posible contexts. The project specific context is defined as a configuration of a context model. The context adapted process is a SPEM 2.0 process model with variabilities resolved.

3.1.1 Organizational Process Model

Process models are defined using SPEM 2.0 [25], the OMG standard for process modeling. Actually we use eSPEM, a subset of SPEM 2.0 that is enough for our experimental purposes.

SPEM 2.0 provides some primitives for specifying variability as shown in Fig. 2. A SPEM compliant complete process model is modeled as a *Method Plug-in* including *Process Elements* and their linked *Method Content Elements*. *Method Content Elements* specifically correspond to *Task Definitions* having *Work Product Definitions* as input and output, and performed by (or participate with) *Role Definitions*. An *Activity* is a *Work Breakdown Element* and a *Work Definition* that define basic work units within a Process as well as a Process itself. An *Activity* supports the nesting and logical grouping of related *Breakdown Elements* forming breakdown structures. The concrete breakdown structure defined in an *Activity* can be reused by another *Activity* via the *used Activity* association which allows the second *Activity* to reuse its complete sub-structure. So, *Role Use*, *Task Use* and *Work Product Use* are *Work Breakdown Elements* that refer to activity-specific occurrences of the respective *Method Content Element*. A *Variability Element* is a SPEM element that can be modified or extended by other *Variability Element* of the same kind according to a *Variability Type* (extends, replaces, contributes, extends-replace). So, each *Method Content Element* (*TaskDefinition*, *RoleDefinition* and *WorkProductDefinition*) and the *Activity* meta-classes are *Variability Elements*.

We use *Variability Elements* to implement alternatives (labeled with an alternative symbol similar to that used in feature models). A set of alternatives can be defined from the same *Variability Element* (maybe abstract). So, when a *Process Element* is linked to the *Variability Element*, one of these alternatives could be selected. For example a *Task Use* can be linked to one of many available and consistent *Task Definitions*. Additionally, each *Work Breakdown Ele-*

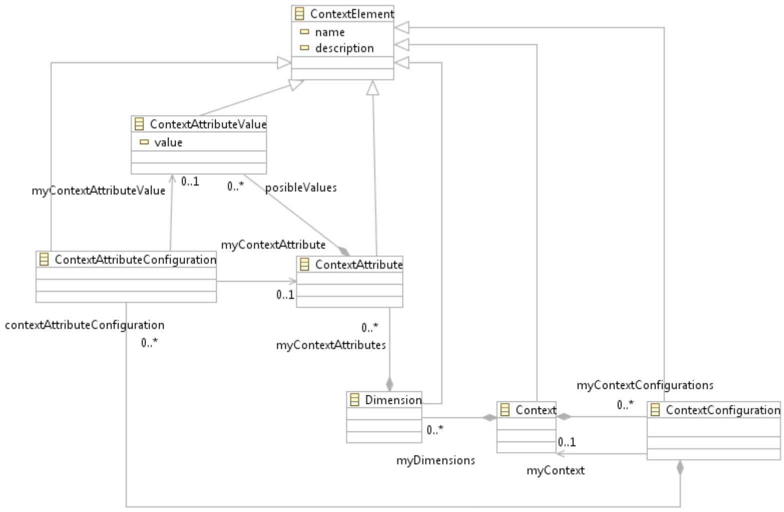


Figure 3: Software Process Context Metamodel - SPCM

ment can be considered as optional or not according to the *isOptional* attribute. Optional elements are labeled with a circle.

Following a general approach for specifying variability in Domain Engineering, we use Feature Models [11] to formalize process variability at a high level of abstraction. We consider software process features as special kinds of software features, such as process properties (life cycle type, maturity level, etc.), method elements (method fragments), process elements (process components, process fragments), process with method elements (chunks) and method plug-in elements (reusable components, processes and configurations). We use the feature model proposed by Czarnecki [11], but using SPEM 2.0 stereotypes.

3.1.2 Context Model

The context of a project may vary according to different project variables along specific dimensions such as: size, duration, complexity, development team size, knowledge about the application domain, or familiarity with the technology involved. Formalizing these characteristics as a model enables us to automatically tailor the organizational process according to them. We have defined SPCM (Software Process Context Metamodel) for defining the context model for each project (see Fig. 3).

SPCM is based on three basic concepts: *ContextAttribute*, *Dimension* and *ContextAttributeConfiguration*. Every element in SPCM extends a *ContextElement* that has a name and a description. A *ContextAttribute* represents a relevant characteristic of the process context required for tailoring. The *ContextAttribute* includes a priority (used when a trade-off between context attributes is required) and it can take one of a set of values defined as *ContextAttributeValue*. An example of a *ContextAttribute* is the Project Size. *ContextAttributeValue* represents a type for qualifying a *ContextAttribute*. Examples of *ContextAttributeValues* for Size *ContextAttribute* are the *ContextAttributeValues* {Small, Medium, Large}. *Dimension* represents a collection of related *ContextAttributes*. A *Dimension* eases the separation of concerns applied to *ContextAttributes*. An example of *Dimension* is Team dimension, referring to team attributes such as team size or team capabilities. A *Context* is represented as a collection of *Dimensions*. A *Context* represents the whole context model. To represent possible specific process contexts, Context Configurations can be defined from the context model. A *ContextConfiguration* is a collection of *ContextAttributeConfiguration* that is set to one of the possible *ContextAttributeValues* for *ContextAttribute*. Therefore, a *ContextAttributeConfiguration* is associated to a *ContextAttribute* and to one unique *ContextAttributeValue*. An example of a *ContextAttributeConfiguration* is the ProjectSizeConfiguration for a small project, where its *ContextAttribute* is Project Size and the *AttributeValue* associated is “Small”.

3.1.3 Project Adapted Process Model

The project adapted process model also conforms to SPEM 2.0 metamodel, but it cannot have variabilities, so all variabilities identified as part of the organizational process model are resolved by the tailoring transformation.

3.2 Tailoring by Model Transformation

We use ATL [18], a declarative language, for defining the tailoring transformation rules. Thus, rules about tailoring the general process model according to the values of different context dimensions can be composed incrementally. In this way we can configure new process models through a generative strategy by recombining partial tailoring transformation rules, and thus reusing the knowledge they embody.

In this MDE approach, the project manager should only provide the characteristics of the particular project at hand, and a process specifically adapted to the project is automatically generated. Thus, all and only the required roles, activities and work products will be present in the adapted process, and no extra work would be required. Therefore, the adapted process is more efficient, and the tailoring process is more reliable as well.

The tailoring transformation is endogenous [10] because its output conforms to the same metamodel as the input. However, it is not *in place* since we want to preserve the organizational process model for future configurations. We

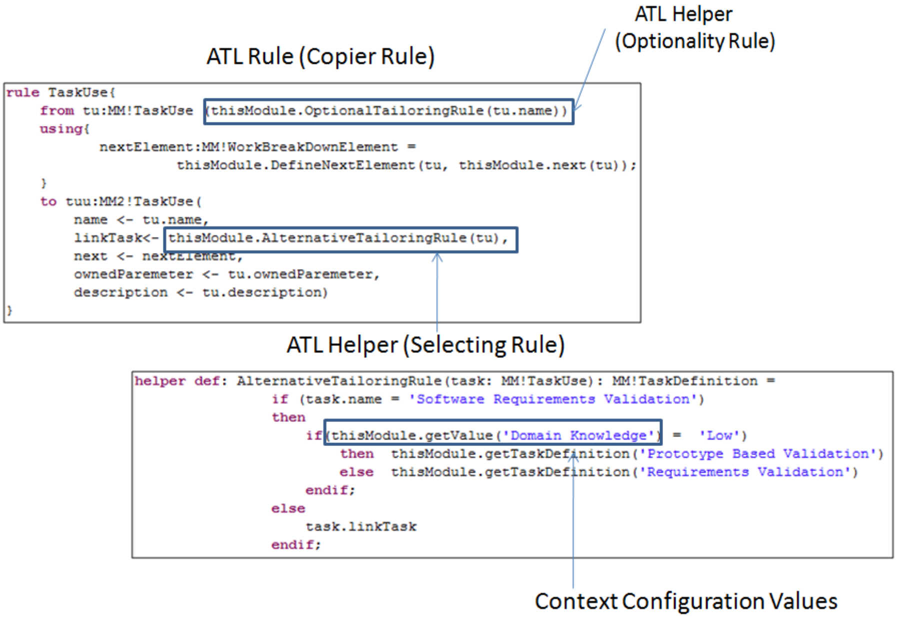


Figure 4: ATL Tailoring Transformation

use *ATLCopier*[1] as a basic template, and we modify it so that only those elements whose rules evaluate to true are actually copied to the target model.

Matched rules constitute the core of an ATL declarative transformation since they allow us to specify: (i) which target elements should be generated for each source element, and (ii) how generated elements are initialized from the matched source elements. In our tailoring rules we make decisions for identified variation points in the process model. Each variation point has an associated helper called from the matched rule. Figure 4 shows rule *TaskUse*. The source pattern *MM!TaskUse* is defined after the keyword *from*, meaning that the rule will generate target elements for each source element matching the pattern. In order to select only those source elements that are relevant for the specific project, an extra condition is added: an Optionality rule implemented as a helper function. When this rule returns false, the element needs to be removed from the process. Attribute initialization uses the values in the source process model element. However, and provided that we use eSPEM variability mechanisms, a process element (e.g. *TaskUse*) could be linked to several variants of method elements (e.g. *Task Definition*). Therefore, we define an *AlternativeTailoringRule* as a rule that returns the selected method element according to the helper rule. The *AlternativeTailoringRule* chooses the most suitable *TaskDefinition* variant, according to the Domain Knowledge Value in the context. If there were more variability points, a conjunction of rules would be applied, also specifying priorities to make trade-offs.

3.3 Tool Implementation

The tool implementation was developed in Eclipse Modeling Framework - EMF 3.4[2] and the ATL plug-in 2.0[3]. Metamodels were defined as ecore metamodels in EMF and the transformations were implemented as ATL rules. Models were implemented as instances of defined metamodels and edited using Exeed (Extended EMF Editor), the reflective editor of EMF. We are currently implementing a plug-in to incorporate the rule transformation-based tailoring approach into EPF. SPEM is being implemented as eSPEM, the experimental version including only the main elements for supporting our approach.

4. TAILORING A REAL WORLD REQUIREMENTS ENGINEERING PROCESS

We have formalized the general requirements engineering process used by a medium size Chilean software company. This company has provided its organizational process as part of the Tutelkán project [34] and it is publicly available [4].

For illustrating our tailoring approach we took the requirements engineering process, along with its adaptation guidelines. These guidelines indicate that certain artifacts should or should not be included as part of the adapted process, according to certain context values. In this way, there are a series of predefined project types such as large development, small development, maintenance or incident. We show how our approach is able to automatically produce the expected process for these project types. We also show how we are also able to produce an appropriate process for an unexpected context as a maintenance without documentation available. All these results have been analyzed and validated by the company's process engineer.

4.1 Organizational Process Model

In the general requirements engineering process we can identify two main components that are executed asynchronously: *Requirements Development* and *Requirements Man-*

[1]ATL Transformation Zoo. `http://www.eclipse.org/-m2m/atl/atlTrnsformations/`
[2]EMF website http://download.eclipse.org/tools/emf
[3]ATL website http://www.eclipse.org/downloads/

[4]Tutelkán: `http://www.tutelkan.org`.

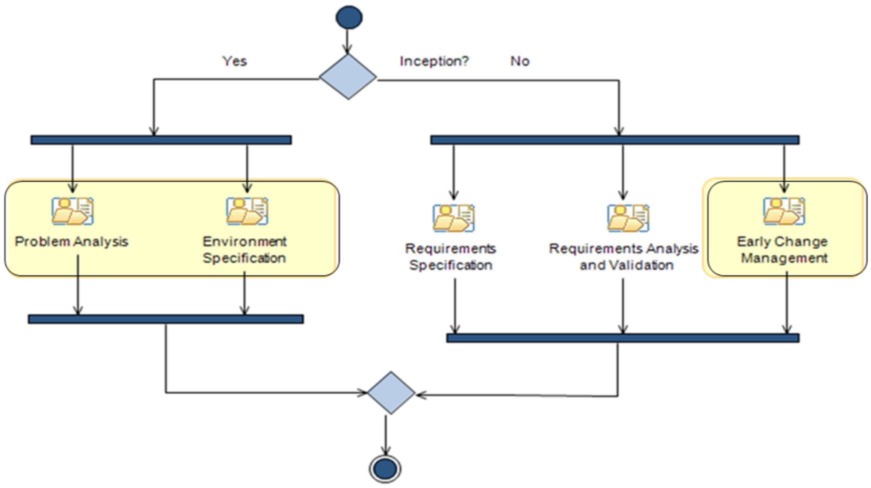


Figure 5: Requirements Development

agement. Figure 6 shows the process formalization in the tool.

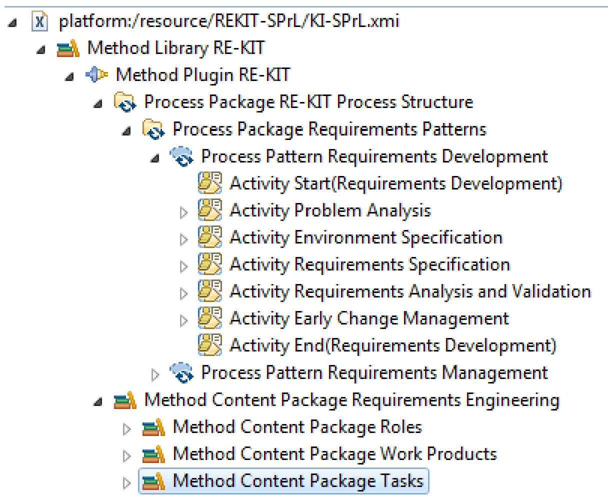


Figure 6: Requirements Engineering Process

Requirements Development is depicted in Fig. 5. Here the process may take two different forms depending on the development stage. In the Inception stage, this process is formed by two parallel and optional activities: *Problem Analysis* and *Environment Specification*. In all other stages, this process is formed by three parallel activities: *Requirements Specification*, *Requirements Analysis and Validation* and *Early Change Management*; only the latter is optional. Also the *Problem Analysis* is formed by the *Preliminary Analysis* and the *Project and Problem Scope Definition*, and this latter one is also optional.

Requirements Management consists of *Requirements Understanding*, *Requirements Commitment*, and then in parallel *Requirements Tracking* and *Requirements Change Management*, as shown in Fig. 7. The *Requirements Understanding* process is illustrated in Fig. 8. It is formed by three tasks: *Identify Requirements Providers*, *Requirements Review* and *Ensuring Common Requirements Understanding*.

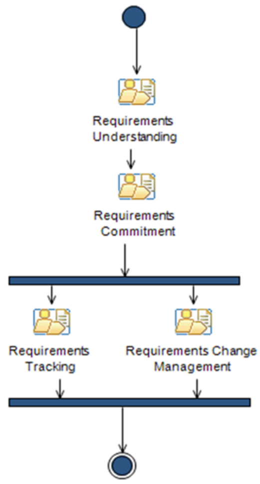


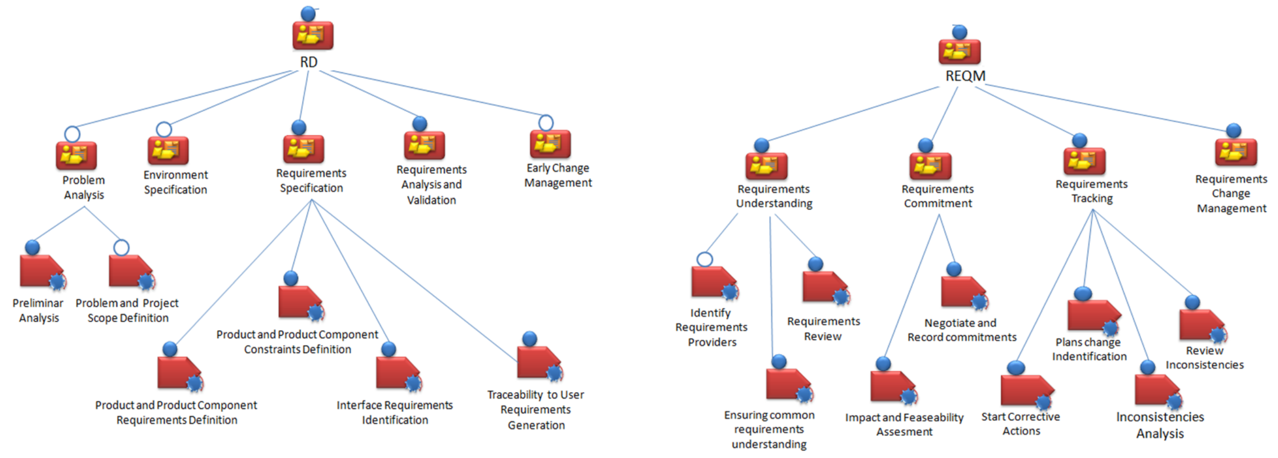
Figure 7: Requirements Management

Notice that the *Identify Requirements Providers* is marked as optional. In this case, the task will only be carried out if the project is a new development.

All optionalities in the process can be summarized in a Process Feature Model [11] as shown in Fig.9.

4.2 Context Model

The general requirements engineering process model presented in the previous section is applied in different kinds of projects. Several dimensions and attributes have been identified as relevant by the company for characterizing projects. Figure 10 shows the context model. The *Domain* dimension has three attributes: *Application Domain*, *Development Environment* and *Source of Documentation*. The first two may be either known or unknown, and the last one may exist, not exist, or there may be an expert who may provide information. Similarly, the *Team* dimension has two attributes: *Team Size* and *Team Expertise*, each one with their corresponding values. The *Management* dimension has five attributes: *Project Type*, *Provider*, *Business*, *Customer Type* and *Project Duration*.



(a) (b)

Figure 9: (a) Requirements Development and (b) Requirements Management Feature Models

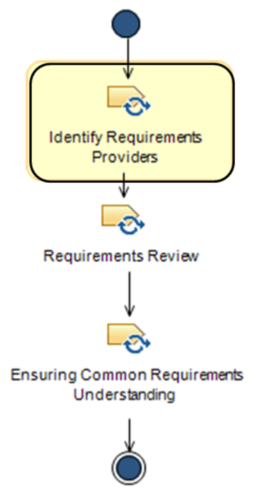


Figure 8: Requirements Understanding

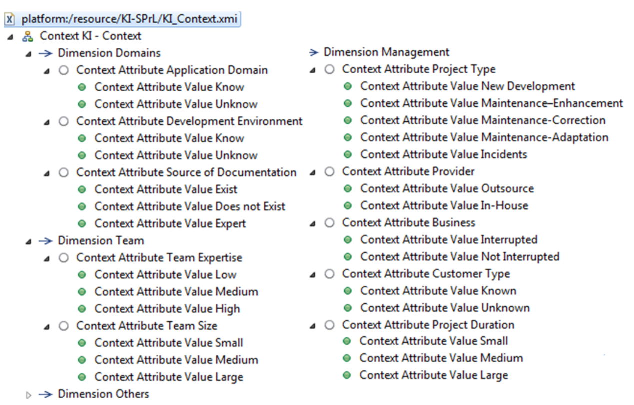


Figure 10: Context Model

The second column in Tab. 1 describes the values of the context variables for a new development within an unknown application domain, whose documentation does not exist, where the development environment and costumer type are unknown, the provider is in house, and the duration is small. In this case the tailored process expected would include all the optional tasks, roles and work products as it is the most complex situation.

Table 1: Two project contexts

Context attribute	Novel Development	Simple Maintenance
Project type	New development	Corrective Maintenance
Application domain	Unknown	Known
Documentation	Does not exist	Exist
Provider	In-house	In-house
Development environment	Unknown	Known
Customer type	Unknown	Known
Project durantion	Small	Medium

On the other hand, the third column in Tab. 1 describes a simple maintenance corrective project, where the application domain, the development environment and the costumer type are known, the documentation exists, the provider is in house and the duration is medium. In this case a much simpler process is expected to be applied. Figure 11 shows both the *Requirements Development* and the *Requirements Understanding* activities where some optional tasks have been removed from the context adapted process.

4.3 Tailoring Transformation

The tailoring transformation takes the general requirements process and a particular context model, and automatically yields a context adapted process. To this end particular rules are provided so that, according to particular values in the context dimensions, decisions could be made about all variation points identified as part of the Feature Model. Table 2 shows some of the directions included in the

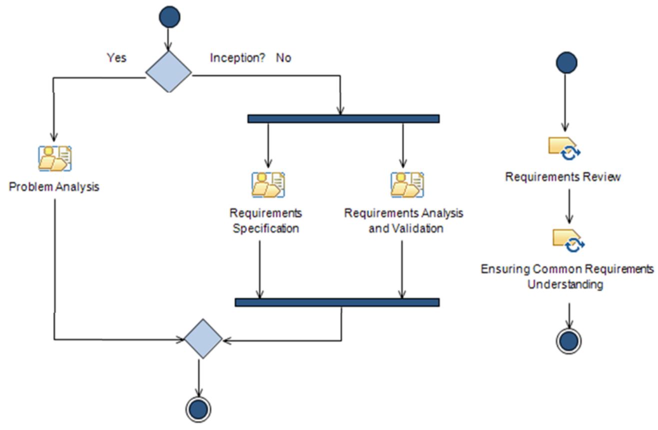


Figure 11: Req. Development and Req. Understanding for a simple Maintenance project

original adaptation guideline that were taken as a starting point for building the transformation rules.

Table 2: Adaptation guidelines

Context attribute	Value	Action
Project type	Maintenance Enhancement	Problem and Project Scope Definition Task is required
Project type	Maintenance Correction	Early Change Management Activity is not required
Provider	In house	Problem and Project Scope Definition Task could be required
Provider	Outsource	Problem and Project Scope Definition Task is required
Source of Documentation	Does not exist	Environment Specification could be required
Source of Documentation	Exist	no action is suggested

It is clear from the table that most common contexts are described and there is no ambiguity about the expected adapted process. For example, for Maintenance-Correction project type, the Early Change Management Activity is never required. However, there are certain combinations of attribute values that are not defined. For example, for Provider in house, the Problem and Project Scope Definition Task could be required or not depending on the values of other attributes, but it is not clearly established. There are still other situations, like that happening when the Source of Documentation exists, where there is no clear action to be taken. Moreover, there are situations (not shown in the table) where the action to be taken does not only depend on the value of one attribute, and if there are two or more attribute values that yield contradictory actions, priorities should be established. In these cases, there is an evident need to rely on a tool that is able to make an appropriate decision by combining partial decisions about different values in the context. In this way evolvability is also achieved since partial rules could be adjusted over time without affecting others.

Figure 12 shows an abstract tree of conditions on attribute values for determining the inclusion of the *Environment Specification* activity and the following code shows the ATL implementation of the rule.

```
–Rule 2 - Environment Specification Activity selection
helper def:activityRule2(elementName:String) : Boolean =
  if (elementName = 'Environment Specification') then
    if (thisModule.getValue('Project Type') = 'Incidents') then
          false
    else
      if ((thisModule.getValue('Project Type') =
            'New Development') or
          (thisModule.getValue('Project Type') =
            'Maintenance-Enhancement') then true
      else
          if (thisModule.getValue('Source of Documentation')
              <> 'Exist') then true
          else false
          endif
      endif
    endif
  else true
  endif
```

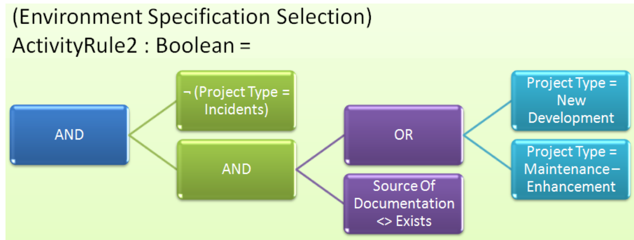


Figure 12: Attribute values for selecting the Environment Specification activity

Let us now consider the case where we have a project context similar to that in the corrective maintenance (third column in Tab. 1), but now considering that the project does not have documentation available. Clearly this is a different case and there is no definition within the adaptation Tab. 2 that indicates the decisions to be made. In this case we configure the project context as shown in Tab. 3, and we apply the rules, in particular Rule 2 just presented.

Table 3: Maintenance without documentation

Context attribute	Attribute value
Project type	Corrective Maintenance
Application domain	Kknown
Documentation	Does not exist
Provider	In-house
Development environment	Known
Customer type	Known
Project durantion	Medium

The obtained process will include the *Environment Specification* that was previously not included provided that the rule indicates that it needs to be included whenever the documentation is not available (see Fig. 13). According to the process engineer, this is the expected result even though it was not explicitly stated in the adaptation guidelines.

The MDE-based strategy was evaluated in a four-hour workshop including business, process and project management people from the company. In this workshop the technical work and a demo of the solution were presented including solutions of past projects and new possible project characterizations. Every possible adapted process was efficiently

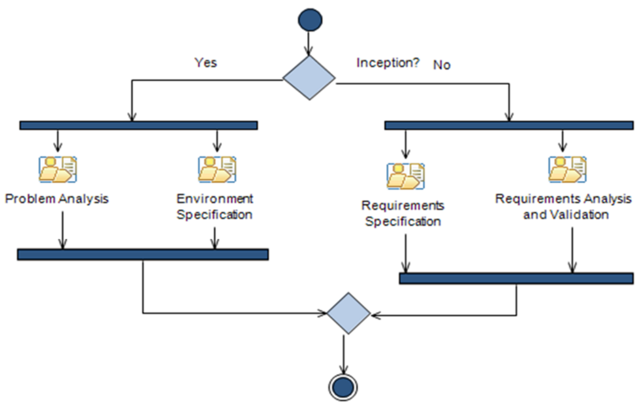


Figure 13: *Requirements Development* process in the case of non existent documentation

generated and collectively evaluated with the process engineer of the host company. The results indicate the generated processes were correct and suitable for each particular project context.

The organizational process was assumed to be already formalized, as well as the adaptation guidelines. The effort involved in generating the formalized organizational process with variabilities was low since it consisted in identifying the process elements affected by the adaptation. Writing the rules was more time consuming mainly because of the inherent ambiguity in the adaptation guidelines. Defining the context model took some time and creativity, but defining a particular context only takes a couple of seconds. Therefore, the return of investment will become more clear as more projects are executed.

5. CONCLUSIONS AND FUTURE WORK

This article proposes a MDE-based strategy for automatically generating processes by tailoring a general process applying a set of transformation rules defined during the organizational process specification. This technique has the potential to improve the project's productivity and quality, as well as the resulting software products. Provided that the adapted process will include all process elements that are required for the particular project context, no extra work will be needed and only the essentially required effort and resources will be spent. In addition, high quality work products can be expected, because the process is adjusted with this goal in each particular project context. Since this tailoring process is automatic and it applies already validated transformations, it is expected to achieve a reduction of the tuning time and cost, and also fewer adaptation errors.

The case study presented in this paper showed that it is possible to apply tailoring transformations built for adapting a general RE process to different project contexts in a planned manner. Being able to validate the transformations for particular known cases has given us confidence on their validity for the general case. Therefore, whenever unanticipated scenarios happen, a combination of already built (and potentially already validated as well) tailoring transformations can be applied; and as a consequence, an appropriate context adapted processes can be obtained quickly and easily. The experience has allowed us to conclude that: (1) our technique is an effective tool to achieve process tailoring and (2) the approach is useful and practical because it was easily implementable by the process group. However, (3) the prototypical tool must be more usable, in particular to define the transformation rules. Additionally, process engineers at the company suggested that the triplet (Context Configuration, Tailored Process and Results) could be saved in order to empirically validate and improve the context model and the tailoring decisions.

We are currently experimenting with this approach in ten other Chilean software companies as part of ADAPTE, a large government funded project. Because of the relevance of the quality of the models in our approach, we have advanced some work designing an analysis framework based on process blueprints [17].

Acknowledgemets

This work has been partly funded by project Fondef D09I1171 of Conicyt, Chile. The work of Julio Hurtado and Alcides Quispe has been also partly funded by NIC, Chile.

6. REFERENCES

[1] A. Aharoni and I. Reinhartz-Berger. A Domain Engineering Approach for Situational Method Engineering. In *Proceedings of the 27th International Conference on Conceptual Modeling, ER'08*, pages 455–468. Springer-Verlag, 2008.

[2] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo. Scoping software process models: initial concepts and experience from defining space standards. In *ICSP'08: Proceedings International Conference on Software Process: Making globally distributed software development a success story*, pages 160–172, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo. Scoping software process lines. *Software Process: Improvement and Practice*, 14(3):181–197, 2009.

[4] X. Bai, L. Huang, and H. Zhang. On scoping stakeholders and artifacts in software process. In Münch et al. [23], pages 39–51.

[5] N. Belkhatir and J. Estublier. Supporting reuse and configuration for large scale software process models. In *Software Process Workshop, 1996. Process Support of Software Product Lines., Proceedings of the 10th International*, pages 35–39, 1996.

[6] B. W. Boehm, B. Clark, E. Horowitz, J. C. Westland, R. J. Madachy, and R. W. Selby. Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, 1:57–94, 1995.

[7] E. Breton and J. Bézivin. Model driven process engineering. In *Computer Software and Applications Conf., 2001. COMPSAC 2001*, pages 225–230, 2001.

[8] D. W. Bustard and F. Keenan. Strategies for systems analysis: Groundwork for process tailoring. In *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, pages 357–362, Washington DC, USA, 2005. IEEE Computer Society.

[9] M. A. Cusumano, A. MacCormack, C. F. Kemerer, and W. B. Crandall. Critical Decisions in Software

Development: Updating the State of the Practice. *IEEE Software*, 26(5):84–87, 2009.

[10] K. Czarnecki and S. Helsen. Feature-based Survey of Model Transformation Approaches. *IBM Systems Journal*, 45(3):621–645, 2006.

[11] K. Czarnecki, S. Helsen, and U. W. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[12] F. Dai and T. Li. Tailoring software evolution process. In *8th ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007.*, volume 2, pages 782–787, 2007.

[13] J. Dörr, S. Adam, M. Eisenbarth, and M. Ehresmann. Implementing Requirements Engineering Processes: Using Cooperative Self-Assessment and Improvement. *IEEE Software*, 25(3):71–77, 2008.

[14] D. Firesmith. Creating a Project-Specific Requirements Engineering Process. *Journal of Object Technology*, 3(5):31–44, 2004.

[15] S. Henninger and K. Baumgarten. A Case-Based Approach to Tailoring Software Processes. In *4th International Conference on Case-Based Reasoning, ICCBR 2001*, volume 2080 of *LNCS*, pages 249–262. Springer, 2001.

[16] J. A. Hurtado and C. Bastarrica. Process Model Tailoring as a Mean for Process Knowledge Reuse. In *2nd Workshop on Knowledge Reuse, KREUSE*, Falls Church, Virginia, USA., September 2009.

[17] J. A. Hurtado, A. Lagos, A. Bergel, and M. C. Bastarrica. Software Process Model Blueprints. In Münch et al. [23], pages 273–284.

[18] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, and P. Valduriez. ATL: a QVT-like Transformation Language. In *Companion to the 21th Annual ACM SIGPLAN Conf. on OOPSLA'2006*, pages 719–720. ACM, 2006.

[19] P. Killisperger, M. Stumptner, G. Peters, G. Grossmann, and T. Stückl. Meta Model Based Architecture for Software Process Instantiation. In *Trustworthy Software Development Processes, International Conference on Software Process, ICSP 2009*, LNCS 5543, pages 63–74, 2009.

[20] S. Koolmanojwong and B. W. Boehm. The Incremental Commitment Model Process Patterns for Rapid-Fielding Projects. In Münch et al. [23], pages 150–162.

[21] P. A. Laplante and C. J. Neill. Opinion: The Demise of the Waterfall Model Is Imminent. *ACM Queue*, 1(10):10–15, 2004.

[22] I. Mirbel and J. Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78, 2006.

[23] J. Münch, Y. Yang, and W. Schäfer, editors. *New Modeling Concepts for Today's Software Processes, International Conference on Software Process, ICSP 2010, Paderborn, Germany, July 8-9, 2010. Proceedings*, volume 6195 of *LNCS*. Springer, 2010.

[24] A. Ocampo, F. Bella, and J. Münch. Software process commonality analysis. *Software Process: Improvement and Practice*, 10(3):273–285, 2005.

[25] OMG. Software Process Engineering Metamodel SPEM 2.0 OMG Beta Specification. Technical Report ptc/07-11-01, OMG, 2007.

[26] L. J. Osterweil. Software Processes Are Software Too. In *9th International Conference on Software Engineering, ICSE'1987*, pages 2–13, 1987.

[27] S. Park, H. Na, and V. Sugumaran. A semi-automated filtering technique for software process tailoring using neural network. *Expert Systems with Applications*, 30:179–189, 2006.

[28] O. Pedreira, M. Piattini, M. R. Luaces, and N. R. Brisaboa. A systematic review of software process tailoring. *SIGSOFT Softw. Eng. Notes*, 32(3):1–6, 2007.

[29] J. Ralyté, R. Deneckére, and C. Roll. Towards a generic model for situational method engineering. In *CAiSE 2003, LNCS 2681*, pages 95–110. Springer-Verlag, 2003.

[30] C. Rolland. Method engineering: State-of-the-art survey and research proposal. In *Proceeding of the 2009 conference on New Trends in Software Methodologies, Tools and Techniques*, pages 3–21, Amsterdam, The Netherlands, 2009. IOS Press.

[31] D. C. Schmidt. Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.

[32] B. I. Simidchieva, L. A. Clarke, and L. J. Osterweil. Representing process variation with a process family. In Q. Wang, D. Pfahl, and D. M. Raffo, editors, *International Conference on Software Process, ICSP'2007*, volume 4470 of *LNCS*, pages 109–120. Springer, 2007.

[33] S. M. Sutton and L. J. Osterweil. Product families and process families. In *ISPW '96: Proceedings of the 10th International Software Process Workshop*, page 109, Washington, DC, USA, 1996. IEEE Computer Society.

[34] G. Valdés, H. Astudillo, M. Visconti, and C. López. The Tutelkán SPI Framework for Small Settings: A Methodology Transfer Vehicle. In *Proceedings of the 17th EuroSPI*, volume 99, pages 142–152, Grenoble, France, September 2010. Communications in Computer and Information Science.

[35] H. Washizaki. Building software process line architectures from bottom up. In J. Münch and M. Vierimaa, editors, *Product-Focused Software Process Improvement*, LNCS, pages 415–421. Springer, 2006.

[36] P. Xu. Knowledge support in software process tailoring. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS '05*, 2005.