Journal of Software: Evolution and Process

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
J. Softw. Evol. and Proc. 2014; **26**:386–403
Published online 8 January 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.1576

MDE-based process tailoring strategy

Julio A. Hurtado Alegría[1,2,*,†], María Cecilia Bastarrica[2], Alcides Quispe[2] and Sergio F. Ochoa[2]

[1]*IDIS Research Group, University of Cauca, Popayán, Colombia*
[2]*Computer Science Department, Universidad de Chile, Santiago, Chile*

SUMMARY

Defining organizational software processes is essential for enhancing maturity because they cannot be improved if they are not specified. However, software process definition is hard and still not good for assuring productivity because the best process depends on the project's particularities. The process engineer can define a specific process for each kind of project, but this is expensive, unrepeatable, and error prone. Moreover, it is difficult to foresee all project scenarios and therefore the appropriate processes. The most usual situation is to apply always the same software process, although it is known to be suboptimal. To deal with this challenge, we propose a model-based approach to software process tailoring that automatically generates project-specific processes on the basis of the organizational process and project contexts. We still require competent process engineers to define the company's process, but once done, our approach is systematic, repeatable, and easy to use. The proposal is applied for tailoring the requirements engineering process of a medium-size Chilean company. Processes obtained matched those used in the company for planned project contexts, and they were also reasonable for nonexpected situations. The company's process and project engineers agreed that the approach was highly valuable.

Received 17 March 2012; Revised 13 August 2012; Accepted 19 October 2012

KEY WORDS: software processes; process 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. But 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 such as CMMI-Dev and ISO/IEC 12207. This organizational process definition always requires a huge effort [1, 2], and it still needs to be adapted to satisfy the characteristics of specific project situations [3].

There is no unique good software process for all projects because appropriateness depends on various organizational, project, and product characteristics [4], and what is even worse, all these characteristics change continuously. For example, if the company develops a new version of an old product, prototyping may not be necessary, whereas it would be a requirement for innovative products. Similarly, an incident project may not need to generate a detailed design work product, whereas it is a requirement for a new development. Therefore, a one-size fits-all approach does not work for software development [5]. Each project has its own characteristics and requires a particular range of techniques and strategies [6], and selecting a set of practices

*Correspondence to: Julio A. Hurtado Alegría, Computer Science Department, Universidad de Chile, Popayán, Chile.
†E-mail: ahurtado@unicauca.edu.co

and integrating them into a coherent process should also be aligned with the business context [7]. It has been suggested that the right set of practices for a project can be better found if we understand the context of the company [8]. We therefore consider that each project context should dictate the definition of the process that best fits it. Moreover, each particular process applied should not vary dramatically from the organizational process, so that process knowledge acquired by the development team could be reused.

Tailoring is the process of configuring a general software process for adapting it to a project's particularities [9]. Empirical studies show that process tailoring is difficult because it involves intensive knowledge generation and deployment [10], and it is also time consuming [11]. Therefore, it is important to reuse the knowledge involved in tailoring so that processes share the same tailoring criteria. Encapsulating this knowledge also allows to evolve it.

Model-driven engineering (MDE) [12] 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 [13], for example, using transformations as instantiation strategies [14]. However, care should be taken in applying MDE because people working in industry are not usually familiar with this approach.

In this paper, we propose an approach for automatically tailoring organizational processes to particular project contexts on the basis of MDE techniques so that appropriate processes are achieved rapidly and with little effort. Tailoring is implemented by means of a series of transformations whose inputs are the organizational process including variabilities, and a model of the project context, and whose output is the context-adapted process, as shown in Figure 1. In a previous work [15], we proposed to manually generate the organizational process model from the organizational process defined by the company's process engineer and also to manually transform the adapted process model back into the process format. These two activities were time consuming and error prone, and the process engineer could hardly understand the software process when it was in its model format, so the approach has been enhanced with automatic transformations from the software processes defined by process engineers to their model version and vice versa. This tailoring strategy is now completely automated in all its steps as shown in Figure 2.

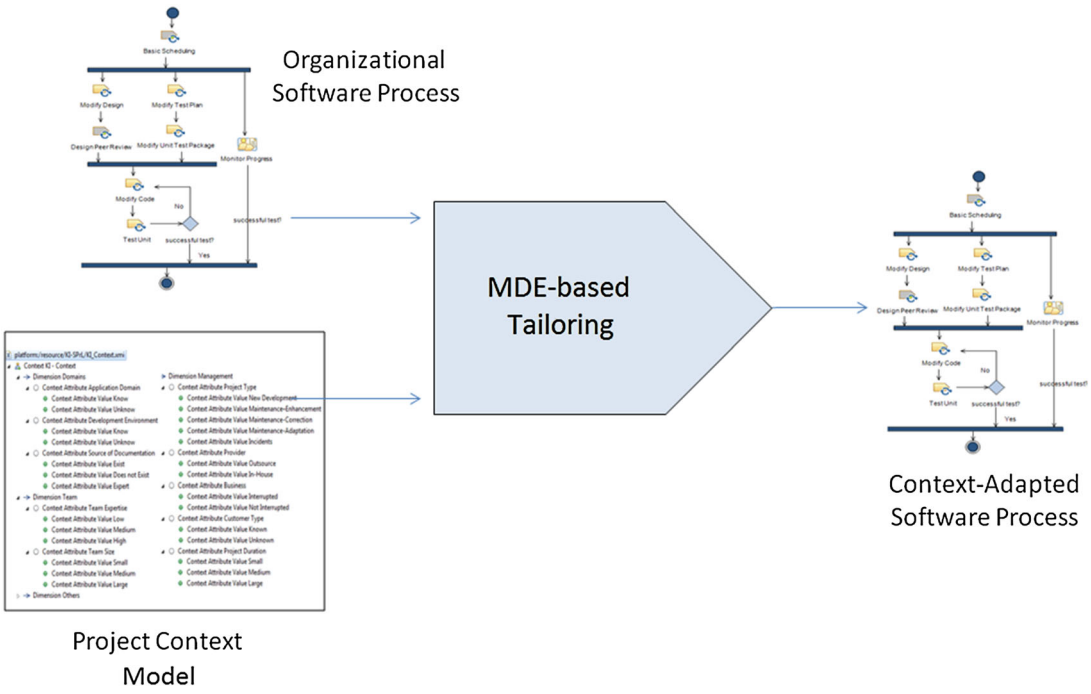

Figure 1. A generative strategy for process tailoring.

DOI: 10.1002/smr

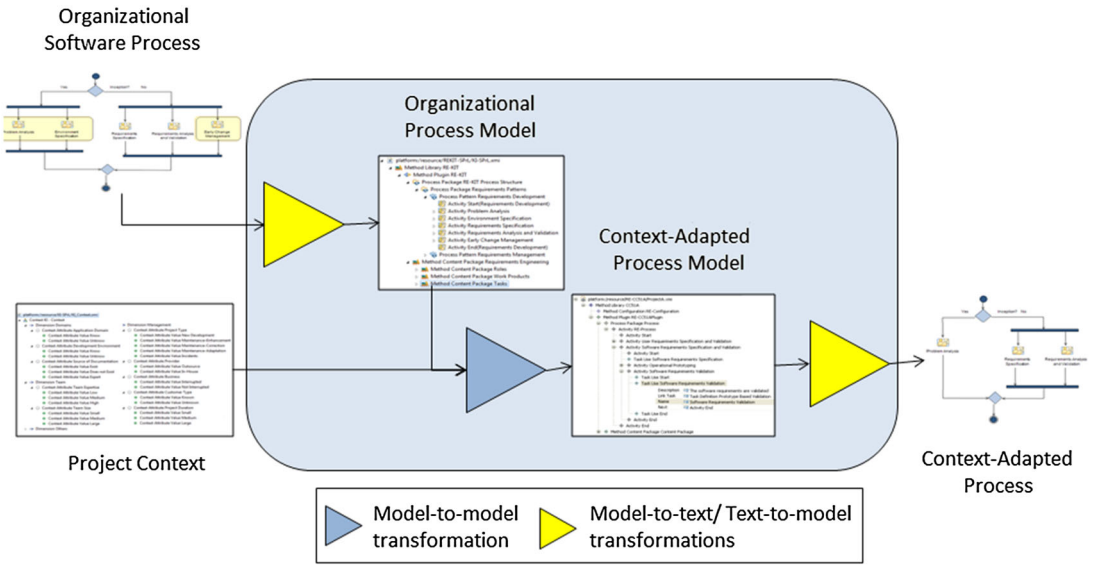

Figure 2. Proposed model-driven engineering approach for process tailoring.

The strategy considers two inputs: the organizational software process and a project context model. The first one is an XML textual representation of the software process as exported by the Eclipse Process Framework Composer (EPFC). The second input is an XML Metadata Interchange (XMI) representation of the context model for a particular project. Provided that the proposed tailoring approach performs a model-to-model transformation, the textual organizational software process is automatically transformed to an XMI model by using a text-to-model transformation. Then, the tailoring transformation, implemented as a set of Atlas Transformation Language (ATL) rules, obtains an adapted process on the basis of the input models. This model-to-model transformation produces an XMI model representing the adapted process. Such a model is then transformed back to an XML representation by using a model-to-text transformation, so that it can be visualized in EPFC. We formalize metamodels by using Eclipse Modeling Framework (EMF) and implement model-to-model transformation rules by using ATL[†] and text-to-model and model-to-text transformations by using Textual Concrete Syntax (TCS).[‡]

Using the proposed approach, this paper illustrates the tailoring of the requirements engineering (RE) process that has been used and evolved for several years in a medium-size Chilean 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.

The paper formalizes the general RE process including its planned variability, and it shows how the proposed MDE-based tailoring approach is actually able to yield the particular process to be followed for each specific context. We were also able to achieve appropriate processes by combining tailoring rules for unanticipated settings. All these results were validated by the company's process engineers and project engineers who found it highly valuable. The strategy was particularly successful because the number of variabilities was not large provided that the process was not large either, so it was possible for the process engineers to understand how each tailoring decision related the project context to each variability point in the process, and they could foresee the result of the rule combination.

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

[†]Atlas Transformation Language (ATL): http://www.eclipse.org/atl/
[‡]Textual Concrete Syntax (TCS): http://www.eclipse.org/gmt/tcs/

DOI: 10.1002/smr

2. RELATED WORK

There are several diverse approaches to tailoring processes. The *assemble approach* [16] enables the implementation of tailoring decisions about deleting and merging process elements. These proposals use formalisms that turn process tailoring a very complex task in practice.

The *situational method engineering* approach focuses on project-specific method construction [17]. During the organizational process definition, an adaptable structure and a guide for process tailoring by situational knowledge are defined [18]. Nevertheless, in most cases, the effort for tailoring the process is huge, especially when an assembly approach is carried out at tailoring time [3]. This is a big problem because process tailoring normally is the responsibility of the project manager but requires the experience and knowledge of the software process engineer, so a suitable separation between their roles is not achieved [19].

Some processes such as the Unified Process [20] use an *adjustment guide approach* where tailoring rules are defined as recommendations to adapt phases, iterations, and disciplines according to project-specific situations. These guides are text written so they are subject to interpretation. This was the approach originally followed in the company where we validated our proposed MDE tailoring approach and we empirically realized that there were some ambiguities, and even people at the company did not agree sometimes about the meaning of certain guides.

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. However, it is difficult, if possible at all, to predict the appropriateness of the process that will emerge. Other processes such as Crystal Methodology [21] 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 [22], 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 sophisticated process engineering knowledge is required to produce a valid configuration each time, whereas in the template strategy, it is difficult to define the adequate set of templates for satisfying each specific project [23].

A *recovery tailoring approach* has been proposed using case-based reasoning [24, 25] and neural networks [26]. 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 are the set-up cost required and the nonplanned change and evolution of various processes, instead of just one. In this case, monotonic knowledge enriches the decision making, whereas new facts that contradict previous experience could result in awkward results.

Killisperger [14] proposed an *instantiation-based approach*. Because the industry has few processes formalized up to the enactment level, this approach may still result in little benefit in practice, but it is promising.

Provided that software processes can be considered as software too [27], a Software Process Line (SPrL) can be considered as a special Software Product Line in the software process engineering domain [15]. SPrL shares common features and exhibits variability [28]. Consequently, an SPrL is an ideal way to define, tailor, and evolve a set of related processes as it is established by the works on process variability representation [29], SPrL architectures [30], process domain analysis [11], and SPrL scoping [31]. An *SPrL approach* facilitates planned reuse, whereas classic tailoring reactively integrates unanticipated variability in the process model [31].

Our work proposes an *MDE tailoring strategy* as a production strategy of project-specific processes in the context of an SPrL.

We use the MDE tailoring strategy using as input an organizational process with variabilities and a specific context model [15]. In this work, the approach has been enhanced with model-to-text and text-to-model transformations, improving thus the understandability of the obtained results.

The context of a software process has been researched in previous work, but it has been usually represented informally. Armbrust *et al*. [32] defined three dimensions to define the characteristics in the SPrL scope definition: product, project, and process. The COCOMO II model [33] defines a set of attributes and dimensions to estimate a project that are useful for representing context models too. The Incremental Commitment Model Process [34] defines a set of patterns for rapid fielding using

DOI: 10.1002/smr

contextualized information. However, these contexts are specific to a process, organization, or research issue. To help organizations determining their relevant dimensions and context attributes, we have defined a Software Process Context Metamodel (SPCM) following the initial ideas presented in [35].

3. TAILORING THE SOFTWARE PROCESS

Defining an organizational software process is necessary if a company wants to improve its development process and completely required to achieve an evaluation or certification such as CMMI or ISO/IEC 12207. Although defining and documenting the process demand an important effort, a general process is still not the best for all projects, even within the same organization. Moreover, an organization that usually develops certain type of projects by using a particular process may eventually get engaged in a different type of project, a new technology, or application domain, and thus the processes that have always worked fine become inadequate. Defining a customized process for each project is too expensive because of 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 presents a promising trade-off.

In the tools implementing each piece of our approach depicted in Figure 2, we have used EPFC [36] to represent the processes in a format that end users are more familiar with. The process variability was defined using a process feature model similar to the one proposed by Kang *et al*. [37]. These specifications generate, through a text-to-model transformation, an experimental SPEM (eSPEM) organizational process model, a process model that is to be tailored through a model-to-model transformation. Finally, the context-adapted process model is used as input of a model-to-text transformation back to XML obtaining a context-adapted process that can be displayed and validated and eventually used by project engineers. So, the MDE tailoring strategy helps achieving a separation between the process modeling stakeholders and process enactment (project) stakeholders [19] and hides the complexity by intensively reusing tailoring knowledge within the transformation chain. Furthermore, the *MDE tailoring strategy* provides a way to cost-efficiently instantiate a general software process into project-specific processes where the project manager should only provide a definition of a specific project context.

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

3.1. Models

All elements in the MDE tailoring strategy are described here and extensively illustrated in the following section.

3.1.1. Organizational software process. Process models are defined using SPEM 2.0 [36], the OMG standard for process modeling, using the EPFC platform and thus conforming to its UMA metamodel.

Following a general approach for specifying variability in Domain Engineering, we use the feature models [37] 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 and process fragments), process with method elements (chunks), and method plug-in elements (reusable components, processes, and configurations). We use the feature model proposed by Kang [37] but using SPEM 2.0 stereotypes.

3.1.2. Organizational process model. We use eSPEM, a subset of SPEM 2.0 that is enough for our experimental purposes, to model the essential parts of the software process model. eSPEM provides some primitives for specifying variability as shown in Figure 3. An eSPEM compliant complete process model is modeled as a *Method Plug-in* including *Process Elements* and their linked *Method Content Elements*. *Method Content Element*s specifically correspond to *Task Definitions* having *Work Product Definitions* as input and output, and performed by (or participate with) *Role Definitions*. An

DOI: 10.1002/smr

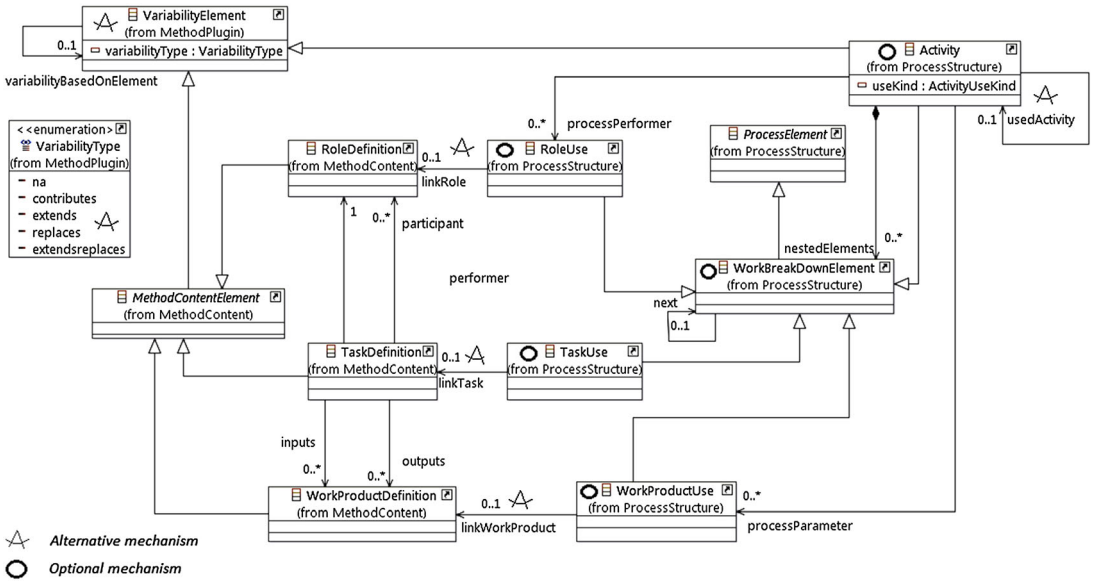

Figure 3. Experimental SPEM highlighting where variability is specified.

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 that allows the second *Activity* to reuse its complete substructure. 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 an eSPEM element that can be modified or extended by other *Variability Element* of the same kind according to a *Variability Type* (extends, replaces, contributes, and extends-replace). So, each *Method Content Element* (*TaskDefinition*, *RoleDefinition*, and *WorkProductDefinition*) and the *Activity* metaclasses 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 the many available and consistent *Task Definitions*. Additionally, each *Work Breakdown Element* can be considered as optional or not according to the *isOptional* attribute. Optional elements are labeled with a circle.

3.1.3. Project 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 for defining the context model for each project (Figure 4). 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 *ContextAttributeValue*s for Size *ContextAttribute* are the *ContextAttributeValue*s (Small, Medium, and Large). *Dimension* represents a collection of related *ContextAttribute*s. A *Dimension* eases the separation of concerns applied to *ContextAttribute*s. 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

DOI: 10.1002/smr

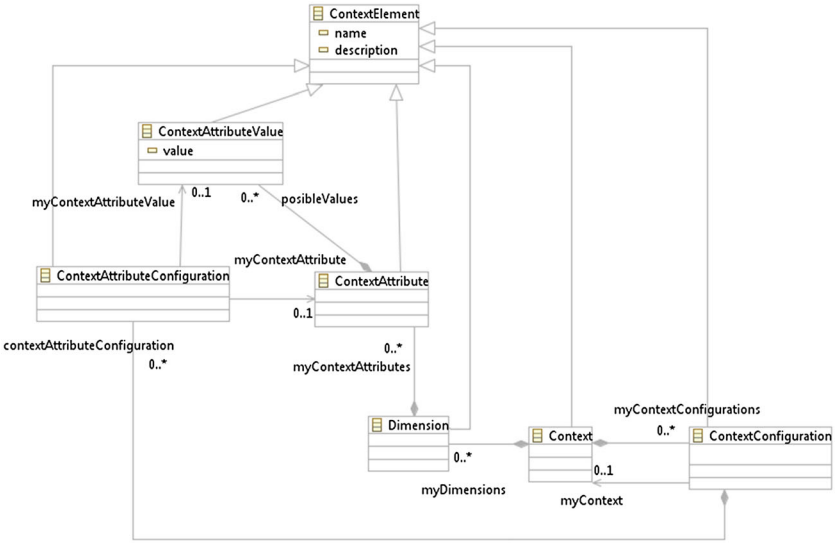

Figure 4. Software Process Context Metamodel.

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*.

3.1.4. Context-adapted process model. The context-adapted process model also conforms to eSPEM, but it cannot have variabilities, so all variabilities identified as part of the organizational process model must be resolved by the tailoring transformation.

3.1.5. Context-adapted software process. This context-adapted process is displayed back in EPFC showing the resulting process in the form of an activity diagram.

3.2. Tailoring transformation chain

3.2.1. Injector: text-to-model transformation. The software process is transformed into a model, so it can be manipulated with model transformation. To do so, an injector is used, that is, a text-to-model transformation that parses the XML generated by the EPFC tool and syntactically transforms it into an eSPEM compliant model. This transformation allows the process users, that is, project managers and software engineers, to deal with graphical representations of the process, which typically are easier to understand.

3.2.2. Tailoring model-to-model transformation. We use ATL [38], 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.

The tailoring transformation is endogenous [39] because its output conforms to the same metamodel as the input. However, it is not *in place* because we want to preserve the organizational process model for future configurations. We use *ATLCopier*[§] 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 because they allow us to specify (i) which target elements should be generated for each source element and (ii) how

[§]ATL Transformation Zoo. http://www.eclipse.org/m2m/atl/atlTrnsformations/

DOI: 10.1002/smr

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.

Optionality rules are implemented as helper functions. When these rules return false, the element needs to be removed from the process.

Attribute initialization uses the values in the source process model element. Because 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 value of certain context attribute. If there were more than one variability point, a conjunction of rules would be applied, also specifying priorities to make trade-offs.

3.2.3. Extractor: model-to-text transformation. Once the adapted process is obtained, it needs to be pretty printed so that the users can validate it and eventually follow it to develop the project for which it was designed. To this end, an extractor is used, that is, a model-to-text transformation that takes the process model as input and syntactically builds a UMA conformant process that can be visualized with EPFC.

3.3. Tool implementation

The tool implementation was developed in EPFC for the organizational software process, EMF 3.4,** TCS for implementing projectors, and the ATL plug-in 2.0 for the tailoring transformation. Metamodels were defined as ecore metamodels in EMF [40], and the transformations were implemented as TCS and ATL rules. Models could be visualized with Exeed (Extended EMF Editor), the reflective editor of EMF.

4. TAILORING A REAL-WORLD REQUIREMENTS ENGINEERING PROCESS

We have formalized the general RE process used by a medium-size Chilean software company that has achieved ISO and CMMI compliance. The company is headquartered in Santiago, Chile. It offers services of web development and design, software development, and consulting in information technologies focused in the financial industry. Currently, the company has an extensive portfolio of more than 90 clients. Since 2004, the company is certified in ISO 9001:2000 for the processes of Development and Software Project Management and Development. Since 2008, it is a CMMI Level 2 certified company, and recently, it was also satisfactorily rated in the following areas of process level 3: Organizational Process Focus, Organizational Process Definition, and Organizational Training. This company has provided its organizational process as part of the Tutelkan project [41], and it is publicly available.††

For illustrating our tailoring approach, we took the RE 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 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 software process

In the general RE process, we can identify two main components that are executed asynchronously: *Requirements Development* and *Requirements Management*.

**EMF website http://download.eclipse.org/tools/emf

††Tutelkán: http://www.tutelkan.info.

DOI: 10.1002/smr

Requirements Development is depicted in Figure 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 Figure 6(a). The *Requirements Understanding* process is detailed in Figure 6(b). It is formed by three tasks: *Identify Requirements Providers*, *Requirements Review*, and *Ensuring Common Requirements Understanding*. 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 [37] as shown in Figure 7. Currently, the organizational software process and its variability are specified separately, but we are experimenting applying weaving models [42] for automatically generating an integrated specification of the process and its variability [43].

Figure 8 shows the process model in EMF as the output of the text-to-model transformation.

4.2. Context model

The general RE 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 9 shows the context model specified using EMF. 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*.

The second column in Table I describes the values of the context variables for a new development within an unknown application domain, whose documentation does not exist, where the development

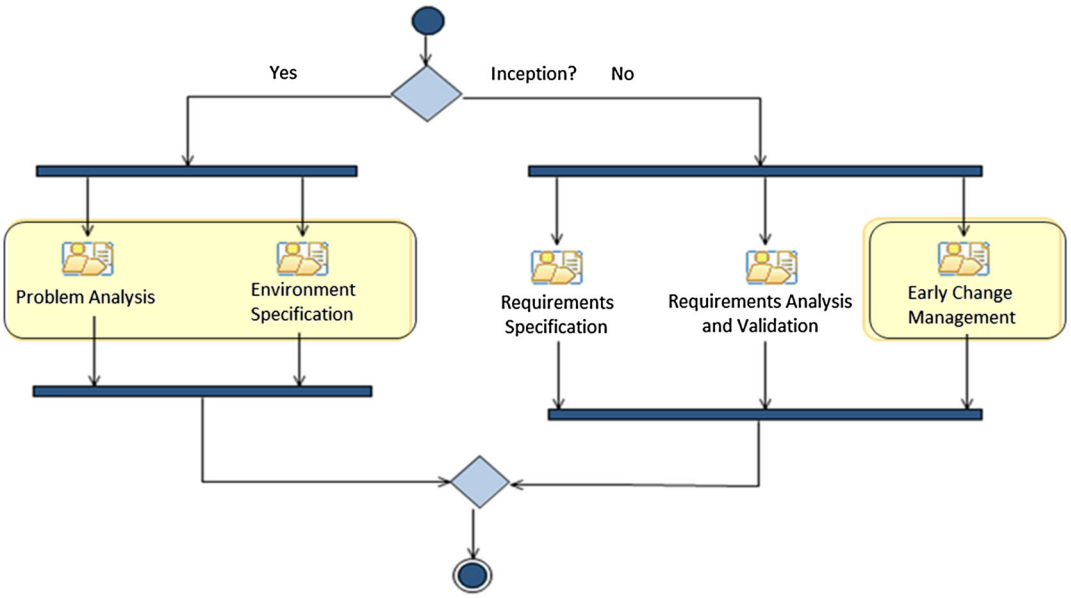

Figure 5. Requirements Development.

DOI: 10.1002/smr

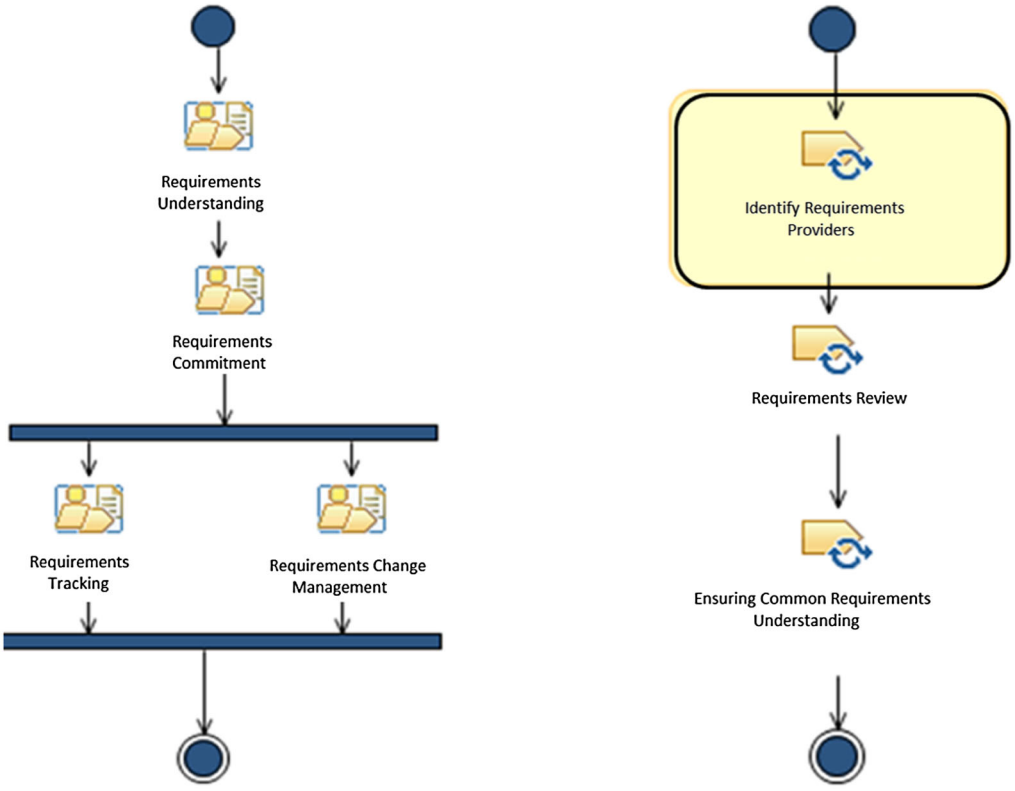

Figure 6. Requirements Management and Requirements Understanding detail.

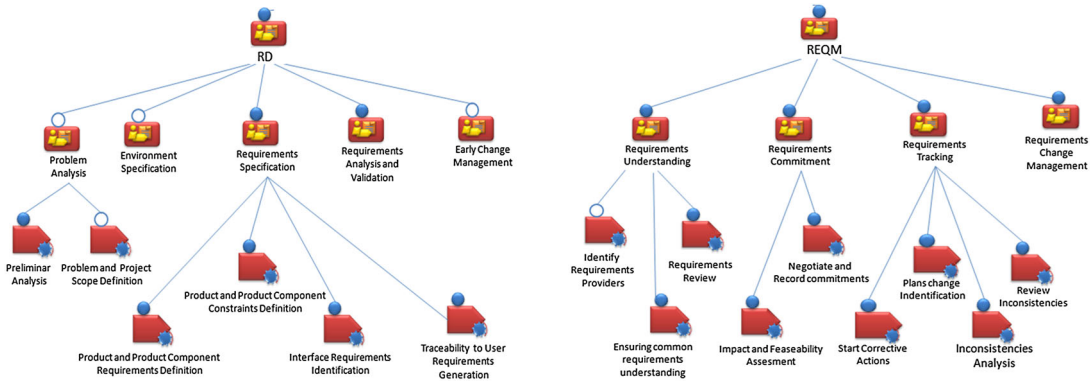

Figure 7. (a) Requirements Development and (b) Requirements Management feature models.

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.

On the other hand, the third column in Table I 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 10 shows both the *Requirements Development* and the *Requirements Understanding* adapted subprocesses where some optional tasks have been removed.

DOI: 10.1002/smr

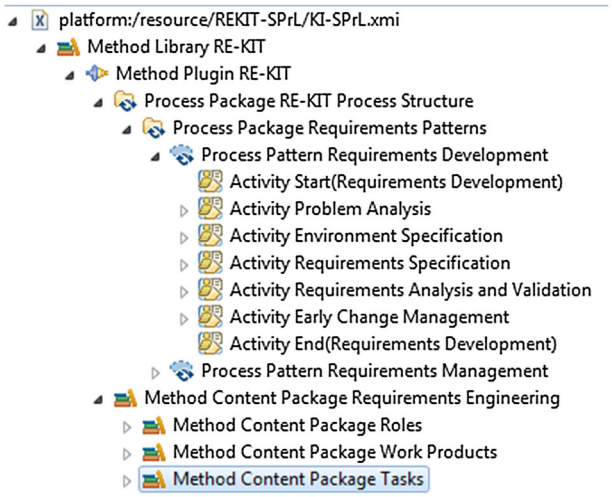

Figure 8. Requirements engineering process model.

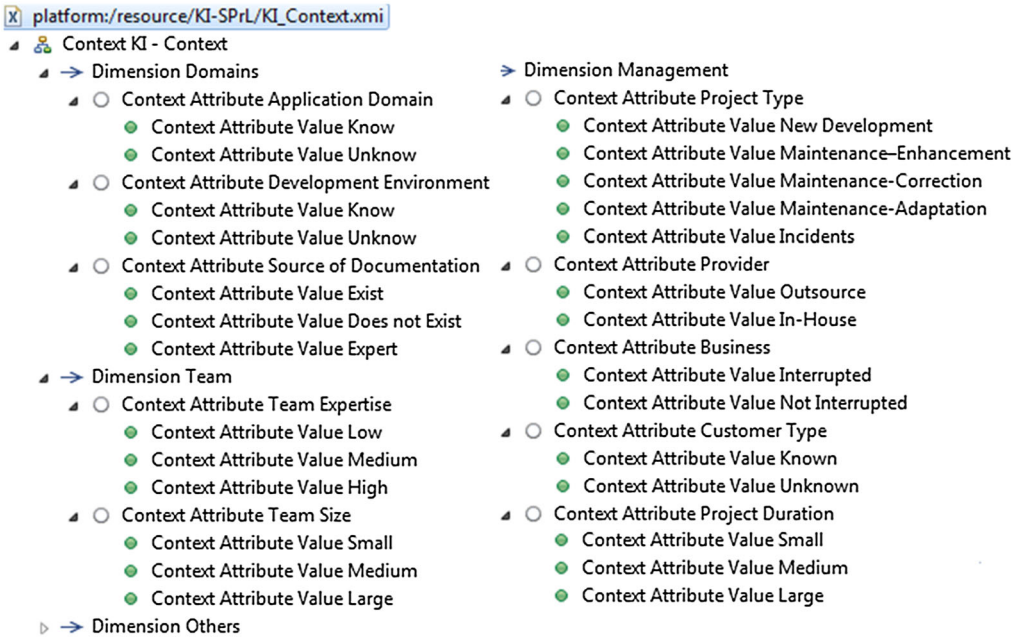

Figure 9. Context model.

Table I. 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 Duration	Small	Medium

DOI: 10.1002/smr

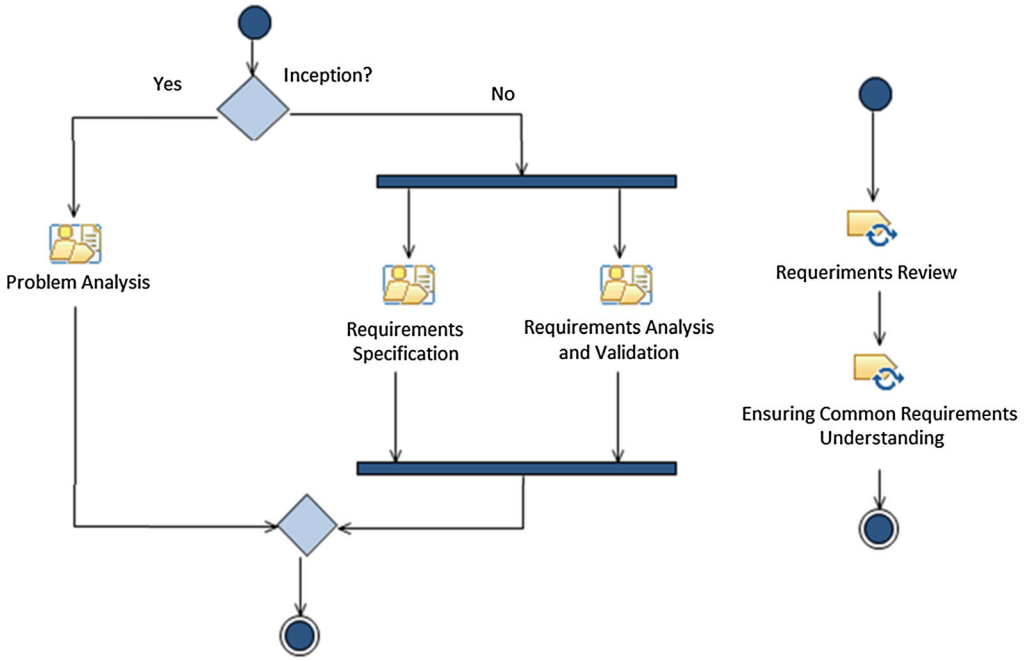

Figure 10. Requirements Development and Requirements Understanding for a simple Maintenance project.

4.3. Tailoring transformation

The tailoring transformation takes the general requirements process and a particular context model and automatically yields a context-adapted process model. 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 II shows some of the directions included in the original adaptation guideline that were taken as a starting point for building the transformation rules.

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, such as 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 supported because partial rules could be adjusted over time without affecting others.

Table II. 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

DOI: 10.1002/smr

Figure 11 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
```

Let us now consider the case where we have a project context similar to that in the corrective maintenance (third column in Table I) 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 Table II that indicates the decisions to be made. In this case, we configure the project context as shown in Table III, and we apply the rules, in particular Rule 2 just presented.

The obtained process now includes 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 (Figure 12). According to the process engineer, this is the expected result even though it was not explicitly stated in the adaptation guidelines.

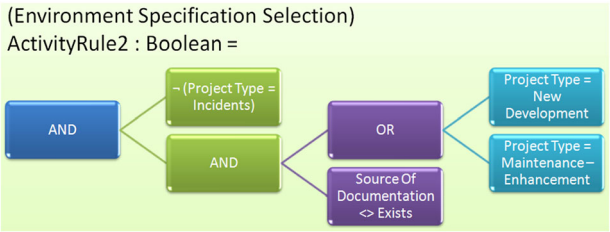

Figure 11. Attribute values for selecting the Environment Specification activity.

Table III. Corrective Maintenance without documentation.

Context attribute	Attribute value
Project Type	Corrective Maintenance
Application Domain	Known
Documentation	Does not exist
Provider	In-house
Development Environment	Known
Customer Type	Known
Project Duration	Medium

DOI: 10.1002/smr

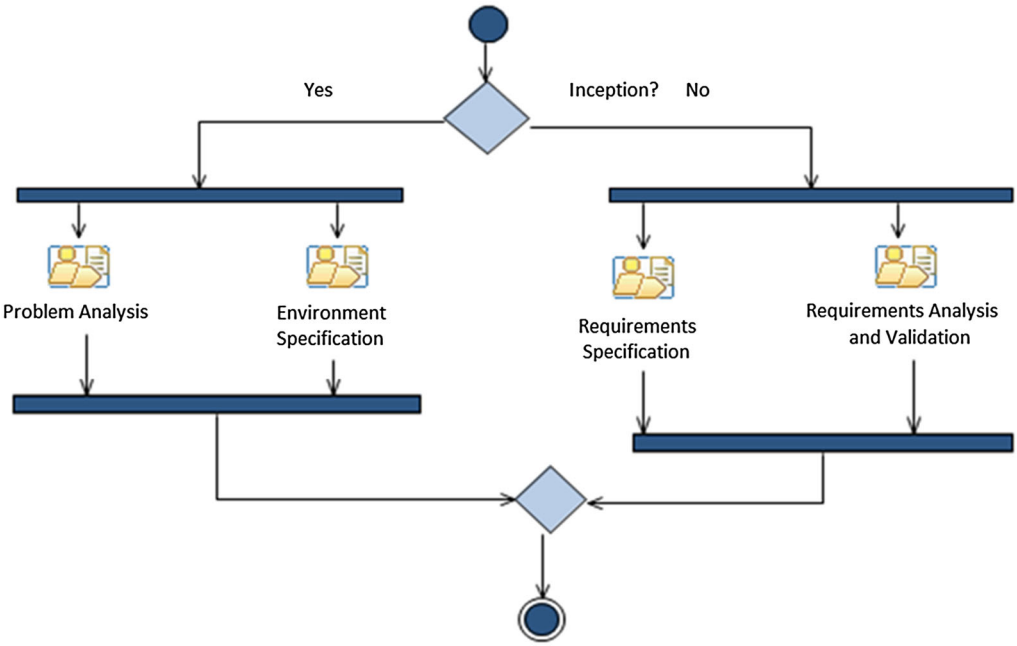

Figure 12. Requirements Development process in the case of nonexistent documentation.

4.4. *Preliminary results*

The MDE-based strategy was evaluated in a 4-h workshop including a business, a process, and a project management person from the company. Every participant had more than 10 years of experience in the software development industry. Particularly, the process engineer and the project manager were the leaders of the organizational software process formalization, and they were familiar with the concepts involved in process tailoring.

In this workshop, the technical work and a demo of the solution were presented including solutions of two past projects and two new possible project characterizations. We counted on the organizational software process already formalized as well as the adaptation rules implemented. In this scenario, the process tailoring was simply reduced to setting the context attribute values. This activity involved between 5 and 10 min for each project. Such a time is used mainly to decide which are the right values for the project context attributes used in the tailoring.

Every possible adapted process was effectively generated and collectively evaluated with the process engineer of the host company. According to the workshop participants, the generated processes were correct and suitable for each particular project context. Although the time and effort spent in these tailoring activities were small, the process engineer highlighted the systematization of the tailoring process as the most valuable aspect of this technique because it makes such a process repeatable and its results are consistent along different projects. The most valuable aspect for the project manager was the fact that he no longer depends on the process engineer to obtain an adapted process for a particular project. The business manager mentions that she envisions a reduction of the development costs provided that the development teams will have now a process tailored to the needs of each project.

The organizational process was assumed to be already formalized, as well as the adaptation guidelines. The effort involved in identifying variabilities in the organizational process was not very high because it consisted in identifying the process elements affected by the adaptation. However, manually transforming the process to its model format was considered not usable; this motivated the inclusion of the injector and the extractor presented in this paper. Defining the context model took some time and creativity, but defining a particular context only takes a couple of seconds. To make this task even more user friendly, we have developed an interactive web-based tool for context model definition and instantiation [44]. Writing the rules was more time consuming mainly because of the inherent ambiguity in the adaptation guidelines. We are currently experimenting in generating

DOI: 10.1002/smr

these rules interactively hiding the complexity of the transformation language as well. Therefore, the return of investment of the whole approach will become more clear as we integrate all the developed tools and more projects are executed.

5. CONCLUSIONS AND FUTURE WORKS

This article proposes an MDE-based strategy for automatically generating processes by tailoring a general process applying a set of transformation rules that consider particularities of project contexts. Provided that the adapted process will include all and only those 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. The tailoring process is automatic, and it applies already validated transformations; therefore, it is expected to achieve a reduction of the development time and cost, and also better quality products. The tailoring strategy is based on a previous work [15] where the process was performed manually.

The case study presented in this paper showed that it is possible to apply tailoring transformations 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 process can be obtained quickly and easily. The experience has allowed us to conclude that (i) the technique is an effective tool to achieve process tailoring, (ii) the approach is useful and practical because it was easily implementable by the process group, and (iii) the prototypical tool became more usable with the text-to-model and model-to-text transformations added. However, it still represents an important limitation for the definition of the transformation rules that need to be written for each company. Additionally, process engineers at the company described in the case study suggested that the triplet (Context Configuration, Tailored Process, and Productivity Results) could be saved to empirically validate and improve the context model and the tailoring decisions.

We are currently experimenting with this approach in six other Chilean software companies as part of ADAPTE,[‡‡] a large government-funded project. Most Chilean software companies are small or medium size; this means that they generally serve a market niche, and thus, their process even though is not necessarily simple, the variations are not many. It remains to be proven that the approach scales for process including a great number of variation points and several context variables. Because of the models' quality relevance in our approach, we have advanced some work designing an analysis framework on the basis of process blueprints [45].

ACKNOWLEDGMENTS

This work has been partly funded by project Fondef D09I1171 of Conicyt, Chile.

REFERENCES

1. Feiler P, Humphrey W. Software process development and enactment: concepts and definitions. *Technical Report* cmu/sei-92-tr-004, 1992. Software Engineering Institute.
2. Garcia J, Rimawi Y, Sánchez M, Amescua A. Ramala: a knowledge base for software process improvement. In *Software Process Improvement, volume 3792 of Lecture Notes in Computer Science*, Richardson I, Abrahamsson P, Messnarz R (eds.). Springer: Berlin/Heidelberg, 2005; 106–117.
3. Mirbel I, Ralyté J. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering* 2006; **11**(1):58–78.
4. Armbrust O, Rombach HD. The right process for each context: objective evidence needed. In Raffo *et al.* [46], 237–241.
5. Firesmith D. Creating a project-specific requirements engineering process. *Journal of Object Technology* 2004; **3**(5):31–44.

[‡‡]ADAPTE project: http://www.adapte.cl

DOI: 10.1002/smr

6. Laplante PA, Neill CJ. Opinion: the demise of the waterfall model is imminent. *ACM Queue* 2004; **1**(10):10–15.
7. Cusumano MA, MacCormack A, Kemerer CF, Crandall WB. Critical decisions in software development: updating the state of the practice. *IEEE Software* 2009; **26**(5):84–87.
8. Dörr J, Adam S, Eisenbarth M, Ehresmann M. Implementing requirements engineering processes: using cooperative self-assessment and improvement. *IEEE Software* 2008; **25**(3):71–77.
9. Pedreira O, Piattini M, Luaces MR, Brisaboa NR. A systematic review of software process tailoring. *ACM SIGSOFT Software Engineering Notes* 2007; **32**(3):1–6.
10. Rolland C. 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*, IOS Press: Amsterdam, The Netherlands, 2009; 3–21.
11. Ocampo A, Bella F, Münch J. Software process commonality analysis. *Software Process: Improvement and Practice* 2005; **10**(3):273–285.
12. Schmidt DC. Model-driven engineering. *IEEE Computer* 2006; **39**(2):25–31.
13. Breton E, Bézivin J. Model driven process engineering. In Computer Software and Applications Conf., 2001. COMPSAC 2001, 2001; 225–230.
14. Killisperger P, Stumptner M, Peters G, Grossmann G, Stückl T. Meta model based architecture for software process instantiation. In Trustworthy Software Development Processes, International Conference on Software Process, ICSP 2009, LNCS 5543, 2009; 63–74.
15. Hurtado JA, Bastarrica MC, Quispe A, Ochoa SF. An MDE approach to software process tailoring. In *International Conference on Software and Systems Process, ICSSP 2011*, Raffo D, Pfahl D, Zhang L (eds.). ACM: New York, NY, USA, 2011; 43–52.
16. Dai F, Li T. Tailoring software evolution process. In 8th ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007, 2007; **2**:782–787.
17. Ralyté J, Deneckére R, Roll C. Towards a generic model for situational method engineering. In *CAiSE 2003, LNCS 2681*, Springer-Verlag: Berlin, Heidelberg, 2003; 95–110.
18. Aharoni A, Reinhartz-Berger I. A domain engineering approach for situational method engineering. In *Proceedings of the 27th International Conference on Conceptual Modeling, ER'08*, Springer-Verlag: Berlin, Heidelberg, 2008; 455–468.
19. Bai X, Huang L, Zhang H. On scoping stakeholders and artifacts in software process. In Münch *et al*. [47], 39–51.
20. Jacobson I, Booch G, Rumbaugh J. *Crystal Clear: The Unified Software Development Process*. Addison Wesley: New York, NY, USA, 1999.
21. Cockburn A. *A Human-Powered Methodology for Small Teams*. Addison Wesley: New York, NY, USA, 2005.
22. Belkhatir N, Estublier J. 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, 1996; 35–39.
23. Bustard DW, Keenan F. 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)*, IEEE Computer Society: Washington DC, USA, 2005; 357–362.
24. Henninger S, Baumgarten K. A case-based approach to tailoring software processes. In *4th International Conference on Case-based Reasoning, ICCBR 2001, volume 2080 of LNCS*, Springer-Verlag: London, UK, 2001; 249–262.
25. Xu P. Knowledge support in software process tailoring. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS '05, 2005.
26. Park S, Na H, Sugumaran V. A semi-automated filtering technique for software process tailoring using neural network. *Expert Systems with Applications* 2006; **30**:179–189.
27. Osterweil LJ. Software processes are software too. In 9th International Conference on Software Engineering, ICSE'1987, 1987; 2–13.
28. Sutton SM, Osterweil LJ. Product families and process families. In *ISPW '96: Proceedings of the 10th International Software Process Workshop*, IEEE Computer Society: Washington, DC, USA, 1996; 109.
29. Simidchieva BI, Clarke LA, Osterweil LJ. Representing process variation with a process family. In *International Conference on Software Process, ICSP'2007, volume 4470 of LNCS*, Wang Q, Pfahl D, Raffo DM (eds.). Springer: Berlin, Heidelberg, Germany, 2007; 109–120.
30. Washizaki H. Building software process line architectures from bottom up. In *Product-focused Software Process Improvement, LNCS*, Münch J, Vierimaa M (eds.). Springer: Honolulu, HI, USA, 2006; 415–421.
31. Armbrust O, Katahira M, Miyamoto Y, Münch J, Nakao H, Ocampo A. Scoping software process lines. *Software Process: Improvement and Practice* 2009; **14**(3):181–197.
32. Armbrust O, Katahira M, Miyamoto Y, Münch J, Nakao H, Ocampo A. 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*, Springer-Verlag: Berlin, Heidelberg, 2008; 160–172.
33. Boehm BW, Clark B, Horowitz E, Westland JC, Madachy RJ, Selby RW. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering* 1995; **1**:57–94.
34. Koolmanojwong S, Boehm BW. The incremental commitment model process patterns for rapid-fielding projects. In Münch *et al*. [47], 150–162.
35. Hurtado JA, Bastarrica MC. Process model tailoring as a mean for process knowledge reuse. In 2nd Workshop on Knowledge Reuse, KREUSE, Falls Church, Virginia, USA, September 2009.

DOI: 10.1002/smr

36. OMG. Software process engineering metamodel SPEM 2.0 OMG beta specification. *Technical Report* ptc/07-11-01, OMG, 2007.
37. Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-oriented domain analysis (FODA). Feasibility study. *Technical Report* CMU/SEI-90-TR-21, Software Engineering Institute, November 1990.
38. Jouault F, Allilaire F, Bézivin J, Kurtev I, Valduriez P. ATL: a QVT-like transformation language. *OOPSLA Companion*, 2006; 719–720.
39. Czarnecki K, Helsen S. Feature-based survey of model transformation approaches. *IBM Systems Journal* 2006; **45** (3):621–645.
40. Budinsky F, Steinberg D, Merks E, Ellersick R, Grose TJ. *Eclipse Modeling Framework: A Developer's Guide*. Addison Wesley: New York, NY, USA 2003.
41. Valdés G, Astudillo H, Visconti M, López C. The Tutelkán SPI framework for small settings: a methodology transfer vehicle. In *Proceedings of the 17th EuroSPI*, volume **99**, Communications in Computer and Information Science: Grenoble, France, September 2010; 142–152.
42. Del Fabro MD, Valduriez P. Towards the efficient development of model transformations using model weaving and matching transformations. *Software and System Modeling* 2009; **8**(3):305–324.
43. Simmonds J, Bastarrica MC, Silvestre L, Quispe A. Modeling variability in software process models. *Technical Report* TR/DCC-2012-3, Computer Science Dept., Universidad de Chile, March 2012.
44. Ortega D. Designing and implementing a tool for software process context configuration. Master's Thesis, Computer Science Department, Universidad de Chile, 2012.
45. Hurtado JA, Bastarrica MC, Bergel A. Analyzing software process models with AVISPA. In Raffo *et al.* [46], 23–32.
46. Raffo D, Pfahl D, Zhang L (eds.). *International Conference on Software and Systems Process, ICSSP 2011, May 21–22, 2011, Proceedings*. ACM: Honolulu, HI, USA, 2011.
47. Münch J, Yang Y, Schäfer W (eds.). *New Modeling Concepts for Today's Software Processes, International Conference on Software Process, ICSP 2010, July 8–9, 2010*. Proceedings, volume 6195 of LNCS. Springer: Paderborn, Germany, 2010.

AUTHORS' BIOGRAPHIES:

Julio A. Hurtado Alegría is an assistant professor of computer science at the University of Cauca, Colombia. In 1997, he obtained his engineering degree from the University of Cauca, and in 2012, he received his PhD degree from the University of Chile. His research interests are in software engineering, particularly formal methods, model-driven engineering, and software processes. Dr. Hurtado Alegría has also worked as a process engineer in several small software companies both in Chile and Colombia.

María Cecilia Bastarrica is an assistant professor at the Computer Science Department at the Universidad de Chile. She coordinates the Model and Transformation Engineering (MaTE) group since 2007. She received the following degrees: PhD in computer science and engineering from the University of Connecticut in 2000, Master of Science from the Catholic University of Chile in 1994, and Bachelor in Engineering from the Catholic University of Uruguay in 1991. Her main research topics are software engineering, software architecture, model-driven engineering, and software product lines. Lately, her work has focused on applying model-driven techniques for modeling software processes.

Alcides Quispe received a degree in systems engineering from the Catholic University Católica Santa María of Arequipa, Peru. After that, he worked for about nine years as a software analyst and software developer in Peru. He is currently a PhD student at the Computer Science Department in the University of Chile. His research interests are requirements engineering and software processes focused on small software companies, object-oriented analysis and design, and object-oriented design patterns.

DOI: 10.1002/smr

Sergio F. Ochoa is an associate professor at the Computer Science Department at the University of Chile. He received his PhD in computer science from the Catholic University of Chile in 2002. His research interests include computer-supported collaborative work, mobile and ubiquitous computing, and software engineering. Dr. Ochoa is a member of IEEE, ACM, and the Chilean Computer Society and sits on the Steering Committee of the Latin American and Caribbean Collaborative ITC Research Initiative (LACCIR).

DOI: 10.1002/smr