

Technical Report TR/DCC-2011-14

Computer Science Department, FCFM, University of Chile

December 23, 2011.

Available at: http://www.dcc.uchile.cl/TR/2011/TR_DCC-20111223-014.pdf

A Context Modeling Language to Support Tailoring of Software Processes

Julio Ariel Hurtado^{1,2}, Sergio F. Ochoa¹, Alcides Quispe¹, Cecilia Bastarrica¹
{jhurtado, sochoa, aquispe, cecilia}@dcc.uchile.cl

¹Computer Science Department – Universidad de Chile - Chile

²IDIS Research Group – Universidad del Cauca - Colombia

Abstract. The suitability of a software process model depends on the specific project context where it is applied. When software process models are tailored, enacted or simulated, unstructured information about a particular project context is required. Therefore it is natural to consider the context model related to a software process as a key element to generate suitable software processes. This paper presents a survey of context modeling approaches for tailoring software processes. It also analyses the stages of the software process engineering for which the modeling approaches are useful. The main objective of such analysis is to identify useful constructs of a context meta-model, which can be used to represent specific project contexts. Based on these constructs a Software Project Context Modeling Language (SPCML) has been proposed. The SPCML constructs and a canonical example are presented and discussed.

Keywords: Software process context, project specific context, software process tailoring, model driven engineering.

1 Introduction

Software processes have been recognized as a critical piece for developing software systems [Hum89]. However, defining and applying suitable software processes demands a great effort. Since there is no a unique software process able to deal with various organizational, project and product characteristics, an adaptable process model is required. Typically the organizational characteristics are considered in its general software process, therefore they do not need to be considered for each particular project. However the project and the product features must be considered in every development, because they usually change with the project. Considering these particularities the general software process should therefore tell us both, how the process varies and when it varies depending on the project and product features.

Each development has its own characteristics and requires a particular range of techniques and practices to be performed [LN04]. In this scenario, selecting a set of process constructs and integrating them into a coherent process is almost mandatory. The resulting development process model must be aligned with the organizational business context [CMKC09].

Each project context should dictate the definition of the process that best fits it. In a variety of process models, contextualized information normally is used to make decisions about how to choose or adapt a software process model to specific situations. Computable contextual representations are typically required to automatically perform this tailoring and also to reduce the uncertainty and effort related to this activity. This reduction is particularly relevant for small and medium sized software organizations because they usually count on few time and economical resources to perform such activity. Therefore counting on a context computable model would facilitate and make automatable the software process tailoring.

The context models are subject of research, because a well designed model is the key to establish the most suitable software process at hand. These models typically take into account a project specific variable, e.g. the business or technological risk, the technological support, the skills of the development team, or any other variable affecting the development process. Moreover, these models mainly address

the context modeling considering just one specific process; typically the one defined as organizational process. In contrast with such approach this work intends to define a generic context model (or meta-model) that allow us to address several software process models.

The literature reports various approaches to guide the selection of a process according to a given project context. These works deal with different kinds of context factors characterizing situations. However, to the best of our knowledge, and with the exception of our previous works, the context model has not been formalized in an intentional way; particularly, to support process tailoring via model transformations. This work defines a language for representing context models and facilitates thus its uses in the ADAPTE project¹.

Next section briefly introduces the concepts behind the software process engineering and the related works. Section 3 presents a study on context modeling based on an exhaustive literature review. Section 4 proposes a domain specific language to define project context models. Such language was named Software Project Context Modeling Language (SPCML), and it is intended for describing software project contexts. Section 5 presents the conclusions and the future work.

2 Process Engineering

Process engineering is the software engineering area involved in defining practices to represent, apply, improve and evolve software process [FH93]. Model-driven engineering (MDE) [Sch06] is a software development approach in which abstract models are defined and systematically transformed into more concrete models, and eventually into source code. MDE has been used in software process engineering [BB01], particularly using transformations as instantiation strategies [KSPGS09]. In such transformations the contextual information of a particular project becomes vital to determine how to tailor an organizational software process to become it suitable for such project.

Figure 1 shows the general MDE strategy proposed in ADAPTE to perform software process engineering. The tailoring process uses two components as input: (1) the *organizational software process model* and (2) the *specific context model*. The first one represents the general strategy defined by the software company to address their projects, and the second one capture the context characterization of the project to address (e.g. complexity, duration and risk level). Both components must be specified in computable representations since the tailoring process would be as automatic as possible. It allows the tailoring process to be predictable, repeatable and low-cost [HBQ11].

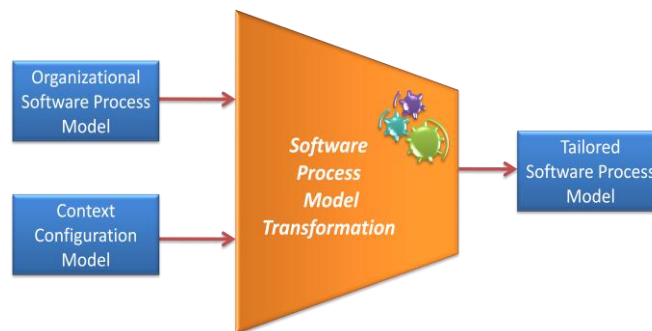


Figure 1. Software process model transformation

¹ <http://www.adapte.cl/>

The *software process model transformation* component uses the two previously mentioned inputs to perform the tailoring process using a set of tailoring rules. The tailoring rules were defined by the software organization according to previous experiences or recognized best practices. The outcome of the *software process model transformation* is the target software process model, also known as the tailored process.

Context process modeling might full fit particular requirements to achieve the target process models from the source process model, for instance the formality level of such process. Additionally, at each stage of the process engineering, context information is about different concerns; e.g. at tailoring, project specific information could be required, at enactment, physical environment information could be need and at simulation statistical information could be required.

3 Study of the Context Modeling

The literature reports a several definitions for context in computer science [BP99, ZLO07]. In case of software development we define the project context as *the set of attributes instances that characterizes the project, the product to be developed, the participating resources, the tools to be used and the environmental conditions*. Context is the element under which the process variability makes sense. An adaptable software process model is insufficient if a context-based mechanism is not used in the adaptation. This section presents an exhaustive literature review that tries to identify context constructs and stages of the process engineering where these constructs are used. Next section describes the framework used to classify these proposals, and then the results of the literature study are presented and discussed.

3.1 Classification criteria of proposals for context modeling

In order to classify the different approaches for using context information in software process engineering, we have defined a set of classification criteria, which are shown Table 1. These criteria has been defined taking into account one of the goals to be reached in the ADAPTE project; i.e. the automatic tailoring of software processes based on project contextual information.

Table 1. Classification criteria for using software projects contextual information

Criterion	Description	Set of the values
Stage	Time point of the process engineering when the context information is used.	{Refinement, Tailoring, Enactment, Analysis, Customization, Simulation, Definition}
Formality	Level of the formality used to define the specific situation.	{High, Medium, Low}
Dimensions	If the concerns taken in count for the situation specification have been grouped by relevant categories.	Yes/No
Utility	Which is the goal to be reached when using the contextual situational information: <i>selecting</i> a process pattern, <i>learning</i> about previous decisions, <i>deriving</i> or <i>configuring</i> a new process model.	{Selection, Analysis, Derivation, Configuration}.
Constructs	Base elements used to specify the situations.	Strings
Representation	It represents the format in which the context model is specified. Such representations go from text to a Domain Specific Language (DSL).	{Text, List, Table, Tree, Graph, DSL}

Next we briefly present the meaning of the non-trivial values that can be assumed by the proposed criteria. It will help understand the analysis presented in sections 3.2 and 3.3. As was indicated in Table 1, the *stage* criterion can take the following values:

- *Refinement*: This stage refers to the stage where a process engineer requires detailing a part of (or the whole) software process model.
- *Tailoring*: It refers to the stage when a software process model must be adapted to be applied to a specific situation.
- *Enactment*: During enactment the software process model is instantiated to a specific development project.
- *Analysis*: This stage refers to the examination of the software process model in order to assess, study or improve its quality.
- *Customization*: During customization the process engineer reuses and adapts a generic or commercial process model in order to make it suitable for a specific organization.
- *Simulation*: This stage represents the execution of a process model on a simulated scenario in order to analyze the process behavior.
- *Definition*: During this stage the specification of an organizational process model is performed.

The level of *formality* of a contextual representation can be:

- *High*: High formality refers to formal specification such as languages, metamodels and ontologies, or any other type of unambiguous representations able to be computed automatically by a software system.
- *Medium*: This category refers to semi-formal specifications, such as data structures and tables.
- *Low*: These are informal specifications, such as textual specifications, which probably will be ambiguous and unable to be processed automatically by a software system.

In order to facilitate the understanding of the values that can be assumed by the *construct* criterion, we will explain more in detail this criterion in the next section using the results of the literature review. The *utility* criterion refers to the use of the context information as support of a certain activity. The values that can be assigned to the *utility* criterion are the following ones:

- *Selection*: In this case the context information is used for selecting a process model from a set of alternatives.
- *Analysis*: In this case the context information is used for analyzing the suitability of a process model to a specific situation.
- *Derivation*: In derivation the context information is used for deriving a context-specific process model from a general software process model.
- *Building*: In this case the context information is used for selecting fragments of a software process model, while the software process model is being built.
- *Configuration*: In this case the context information is used for defining a specific configuration of a configurable process model.

The type of *representation* refers to the specific mechanisms to specify context information. The contextual information can assume the following values:

- *Text*: The representation attribute is set as *text* when the context information is expressed using natural language.
- *List*: The representation is labeled as *list* when it is expressed using a list of typified values, normally pairs <context attribute, value>
- *Table*: The representation is labeled as *table* when it involves the use of tables with context attributes and possible values.
- *Tree*: A value of *tree* is used when the context information is organized by hierarchies, where usually the last level corresponds to values of context attributes.
- *Graph*: Here the context information is organized as *graphs*, where each context construct is defined as a node and the relationships between these constructs are defined as edges.
- *DSL*: Here the context information is defined as an instance of a context domain specific language (Context DSL) using formally the constructs defined in that language.

Using these criteria and also the results of an exhaustive literature review, we have classified the several proposals. The obtained result is shown in sections 3.2 and 3.3.

3.2 Literature review in software project context information

The context of a process varies according to the values assumed by, e.g., different project, team and environmental variables. Due these characteristics are defined (or at least known) for every development, it enables us to automatically transform a software process model from a stage to another. In order to analyze how context information has been represented in previous proposals, we have reviewed and classified the proposals according to the criteria defined in the previous section.

As a starting point we have conducted an incremental literature review about how the context information is used in the stages of the process engineering. This review has allows us to understand different aspects of the context information required at specific decision points. The obtained results are presented in the Table 2.

Table 2. Classification of related works based on the process engineering stages

Proposal	Stage	Formality	Dimensions	Utility	Constructs	Representation
[HWBBK05]	Customization	Low	No	Selecting	Characteristic, Project Capability	Text
[RC99]	Definition, Tailoring and Simulation	Medium	Yes	Selecting, Derivation	Factor, Value	Table
[BCHWMS95]	Enactment (Estimation)	Low	Yes	Other (Planning)	Scale Factor, Feature	Text
[MW06]	Tailoring	High	No	Selecting	Factor, Value	List
[Xu05]	Tailoring	Low	Yes	Selecting, Derivation	Challenge Category, Evaluation Question	Text
[WSW07]	Definition, Refining	Low	No	Refining	Characteristic	Text

Proposal	Stage	Formality	Dimensions	Utility	Constructs	Representation
[KL05]	Tailoring	Low	No	Selecting	Project Problem, Failure Factor	Text
[KSPBKL08]	Tailoring	Medium	Yes	Derivation	Category, Value, Multiplicity	Table
[ST06]	Analysis	Low	No	Selecting	Organization Detail	Table
[GWJAR09]	Enactment	Low	No	Configuration	Question, Answer	Text (Questionnaire)
[MKSPM08]	Customization	Low	No	Selecting	Decision Card (Agile)	Text
[MD07]	Customization	Low	No	Selecting	Project Parameter	Table
[BK05]	Analysis, Customization	Low	No	Analysis, Selecting	Focus, Time-dependent Goal	Text
[HB09]	Definition, Tailoring	Medium	Yes	Derivation	Context Dimension, Attribute, Value, Priority	Context DSL
[KSPGS09]	Enactment	Low	No	Derivation	Constraint	List
[RF09]	Tailoring	Medium	No	Building	Context Situation, Intention	Process DSL
[ZFH05]	Customization	Low	No	Selecting, Building	Organizational Needs	Text
[He02]	Customization	Low	No	Selecting, Building	-	Text
[AKMMO9]	Analysis, Definition, Tailoring	High	Yes	Building, Analysis	Scope, Attribute, Value, Priority, Features, Map	List
[FMZ06]	Enactment	-	-	Derivation	-	List (of rules)
[BR87]	Tailoring	Low	Yes	Building	Goal, Sub Goal, Question	Text
[Lo96]	Tailoring	Low	Yes	Building	Project Requirement	Text
[KB10]	Tailoring	Medium	Yes	Selecting	Driver, Relevance	Table
[PNS06]	Tailoring	Medium	No	Derivation	Project Environment, Parameter	List
[MPDG06]	Tailoring	Low	No	Selecting, Building	Criteria	Text
[RK00]	Simulation	Low	No	Analysis	Condition	Text

In this literature analysis we have grouped the values of the criterion *constructs* according to the software process context meta-model SPCM proposed by Hurtado et al. [HB09]. Such proposal considers several granularities for the contextual information: context dimensions, attributes, and attribute values and priorities. Next we define each one, from the coarsest to the finest of such information.

- *Context dimensions*: It represents a group of the related contextual attributes. The research works presented in Table 2 have called to these contextual dimensions with several names, such as *categories* and *scale factors*.
- *Context Attribute*: It represents a particular contextual element inside a dimension. Similar to the context dimension element, the authors of the reviewed articles have assigned various names to these context attributes, e.g. *characteristic*, *factor*, *driver*, *question*, *criteria* and *constraint*.
- *Context Attribute Value*: It express a specific value to be assumed by a context attribute. The values of an attribute have been called *factor value*, *criteria value* and *answer*.

- *Priority*: It defines a possible priority level to each context attribute. *Relevance* is a possible synonym of priority. *Driver* could be also used instead of priority, if a driver is the strongest context attribute.

There are also relevant constructs that do not have been identified in Hurtado et al. proposal [HB09], and that have been found in the literature review. This is the case of *goals*, *intention*, *multiplicity* and *mapping*, which represent interesting concepts to specify project contexts. The project goals and intention could give higher level information to make decisions at tailoring time. Multiplicity could help us to express replication of process elements at enacting time. Mapping could help determinate the relationships between context attributes, in order to establish priorities, identify drivers and define abstract transformation rules. Of course there are values for the criterion *constructs* that are absolutely ad hoc for a particular situation or projects niche. That is the case of constructs with values such as *project requirement*, *condition*, or *organization detail*. Next subsections present some statistics indicating, from the literature review, the percentage of proposals focused in a certain process stage, goal (or activity to be supported) and formalism level.

3.2.1. Software context representations according to the process engineering stages

Each stage in which the contextual information is used has its own concerns; however a unique meta-model could have sufficient expressive power to deal with them. At each stage the transformations require specific information to create a possible target model.

The activity more demanding for contextual information is the *tailoring* (Figure 2). However *customization* from Commercial Off-The-Shelf Process (COTS Process) has also an important relevance, particularly for selecting a process from a set of commercial options for an organization or specific project.

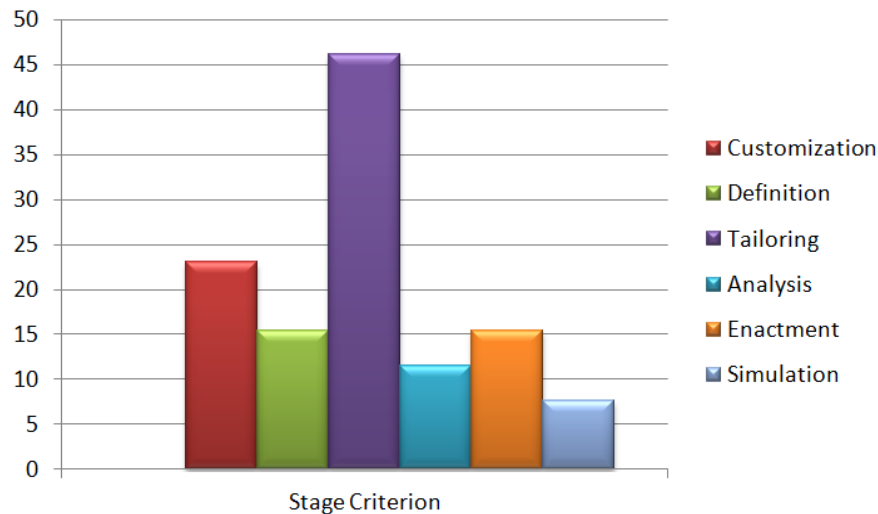


Figure 2. Use of context information by stage of the process engineering

The use of contextual information is also important as support of the software process *definition* and *enactment*. However during the process enactment in real projects, little context information is used by people in charge of such activity.

3.2.2. Software context representations by activity to be supported

The Figure 3 presents how the context information is used according to the literature review. Most works use contextualized information for *selecting* a process model or a software process pattern. However, an important number of works show that such information is also frequently used for *deriving* and *building* a software process model. In these cases the definition of the project context is typically more formal than the specifications used for selecting a process model. Just few works report the use of contextual information for *configuring*, *analyzing* and *refining* software processes.

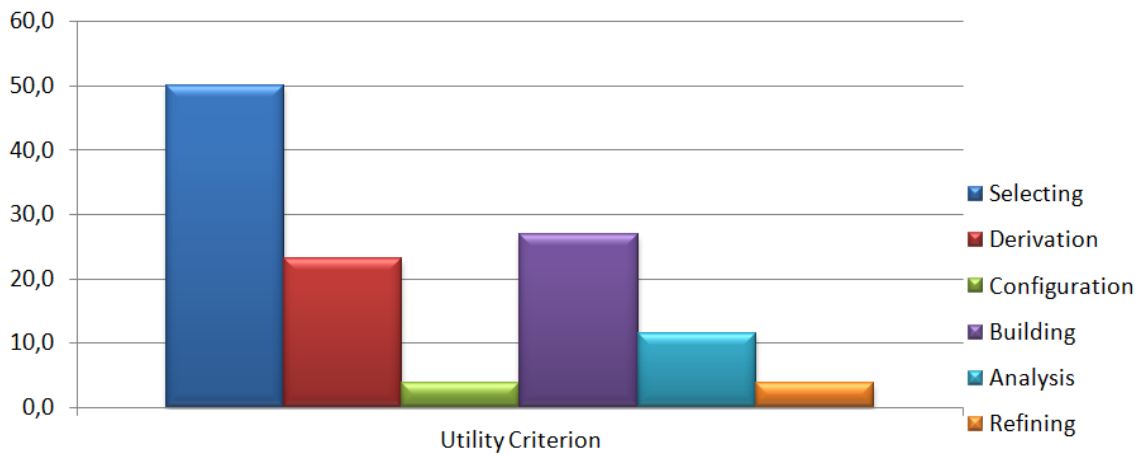


Figure 3. Use of context information by utility criterion

3.2.3. Software context representations by formalism level

Context requires formalization if we want to use it as input for automatic transformations of a software process. Considering the performed literature review, Figure 4 shows the percentage of mechanisms used to represent context information. Only a 7,7% of these representations, i.e. those using a *DSL* specification, are useful to conduct a MDE approach. *Lists* and *tables* are representations near the ideal representation; i.e. these mechanisms could be rich enough to be re-written in a context meta-model.

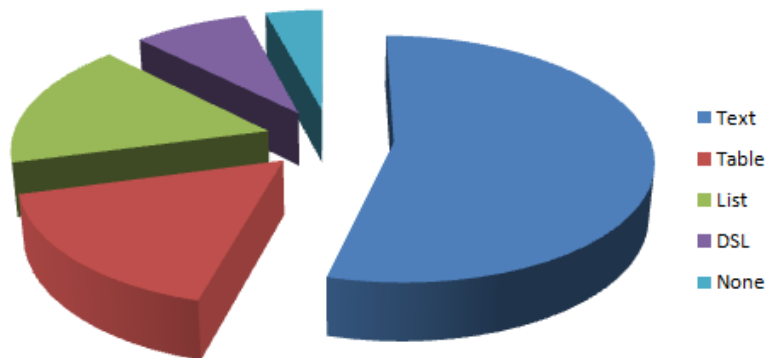


Figure 4. Context Representation

The 53,8% of the proposals use natural language; i.e. *text*. This type of representation is hard understand not only by people but also by MDE applications, and therefore it is not recommended as input to define a unified meta-model, although it offers rich information about the possible constructs (or synonymous)

to give a rich semantic to the meta-model, for instance: criteria, driver, characteristic, question, goal, attribute, factor, feature to define a specific aspect in the context and value, answer, and factor value to specify specific values in a specific situation. Additionally, appear keywords as priority when all the aspects do not have the same weight, challenge as goal to achieve, condition to specify restrictions and maps to related these aspects.

3.3 Languages for Context Modeling

Context models are transversal components, therefore they have been applied to several areas [BD05]. A software process could be specified as a context-aware system, because it is necessary to adjust its definition or behavior in the time, for instance, when the process is tailored to a specific project context. Strang and Linnhoff-Popien [SL04] present a classification of proposals according to the data structures they use to exchange contextual information in context-aware systems. Based on such work, next we briefly explain some of the most relevant proposals.

Key-value pair model: This is the most simple data structure for modeling contextual information. Already Schilit et al. [SAW94] used key-value pairs to model the context by providing the value of context information to an application as an environment variable. In particular, key-value pairs are easy to manage, but it does not have capabilities for sophisticated structuring that enable to use efficient context retrieval algorithms.

Markup scheme models: Similar to most markup scheme modeling approaches, this one involves a hierarchical data structure consisting of markup tags with attributes and content. In particular, the content of the markup tags is usually recursively defined by other markup tags. Typical representatives of this kind of context modeling approaches are profiles. They are usually based on a serialization of a derivative of Standard Generic Markup Language (SGML).

Graphical models: A well-known general purpose modeling language is the *Unified Modeling Language (UML)* which has strong graphical components (UML diagrams). Due to its generic structure, UML is also appropriate to model the contexts. Particularly, in Object-Role Modeling (ORM), the basic modeling concept is the *fact*, and the modeling of a domain using ORM involves identifying appropriate fact types and the roles that entity types play in these. Henricksen extended ORM [HJM05] to allow fact types to be categorized, according to their persistence and source, either as *static* (facts that remain unchanged as long as the entities they describe persist) or as *dynamic*. The latter ones are further distinguished depending on the source of the facts as either *profiled*, *sensed* or *derived* types. Another quality indicator introduced by Henricksen is a history fact type to cover a time-aspect of the context. The last extension to ORM made by Henricksen for context modeling purposes are fact dependencies, which represent a special type of relationship between facts, where a change in one fact leads automatically to a change in another fact: the *dependsOn* relation.

Object-Oriented Model: Similar to other object-oriented context modeling approaches, this one also intends to use the construct provided by this paradigm, e.g. encapsulation and reusability, to cover parts of the problems arising from the dynamics of the context. The details of the context processing are encapsulated into an objects level, and thus such processing is kept hidden to other components. Access to contextual information is provided through specified interfaces only. The context is modeled as an abstraction level on top of the available objects, providing contextual information through their interfaces, hiding the details of determining the output values.

Logic-Based Model: Logic defines the conditions on which a concluding expression or fact may be derived (a process known as reasoning or inference) from a set of other expressions or facts. In order to describe these conditions in a set of rules, a formal system is used. In a logic based context model, the context is consequently defined as facts, expressions and rules. Usually contextual information is added to, updated in and deleted from a logic based system in terms of facts or inferred from the rules in the system respectively.

Some approaches for context modeling has been identified in the SiME - Situational Method Engineering area [KDC10]. Next we describe the most important ones.

Reuse frame: A set of the *criteria* allows specifying a context of method fragments reuse for searching or comparing in order to find an alternative fragment to a used one. The frame includes a reuse situation organized into three dimensions (i.e. organizational, technical and human) and reuse intention.

Interface: A method fragment has associated an interface including context information, such as a *situation* and an *intention*. The situation defines as the method fragment could be used (work products associated) and the intention defines the goal that the method fragment helps to achieve.

Method Service Context: Aims at describing the suitable situation to a project development for a method service (it includes de service proposal). This model includes domain characteristics (e.g. project nature, and project domain) and human (actor), process and product ontologies.

Contingency factors: Context characteristics are described as contingency factors. These factors are used for describing specific projects by assigning values to them.

Development situation: Situations are used to characterize the specific projects and to select configuration packages (methods fragments). The situation model includes a set of the characteristics.

For software processes, the context of a project may vary according to different project variables such as: product size, project duration, product complexity, team size, application domain knowledge, and familiarity with the involved technology, among others. The study case of Perez et al. [PEM96] defines the characteristics of a software process model and its environment, and determines how congruent the process model is in the given environment using project information of a specific organization.

On the other hand, COCOMO II [BCHWMS95] defines a cost model for projects based on situational information. This information is defined as a fix set of factors and their scaling factors. The context dimension is introduced by Mirbel & Ralyte [MR05] as a way to separate context concerns in a matching strategy for selecting chunks in a roadmap approach to SiME.

In the work of Bucher et al. [BKW06] a context engineering method is proposed where context factors are identified and analyzed to enable the engineering of contextual methods. Royce [Roy98] presents an approach to tailor the Unified Process based on two dimensions of characteristics (technical and management complexities). The characteristics in the literature cited above are discriminating factors

such as scale, stakeholder cohesion or contention, process flexibility or rigor, process maturity, architectural risk and domain experience.

According to Aharoni & Reinhartz-Berger [AR08] a specific context can be defined as a vector of characteristics that relate to the organization, the project, the developing team, the customer, etc. However, defining the context as a formal model enables us to automatically tailor the organizational process according to it.

3.4 Contextual variables to be considered in a project context

There are few proposals indicating the context dimensions and variables that can be used to adapt software processes. Antunes et al [ACG10] propose a context model that identifies four dimensions: domain, organization, project and personal. However, the proposal does not include a description of the variables and the values of those variables for such context dimensions.

Araujo et al propose a set of context dimensions and attributes to be considered when defining or adapting software processes [ASRB04]: domain, organization, project, task, team, roles, product, business domain, client/user. Provided that these dimension were not proposed for Small and Medium-sized Enterprises (SME), the number of context dimension overcome the numbers that can be handled by small organizations. In addition, the granularity of the context dimensions is heterogeneous; therefore it is complex to use them as input for process tailoring rules.

Maffin proposes a contextual framework that influence the software development life cycle [Ma98]. It is composed of four context dimensions (project, organization, product and personnel) and fifteen context variables (Figure 5.a). Although the size and scope of the context framework seems to be appropriate for a SME, the context variables are still coarse-grained. It means these variables must be derived in several fine-grain definitions to be used as input of process tailoring rules. Once done this task, final list of variables would probably be too long to be used by SME.

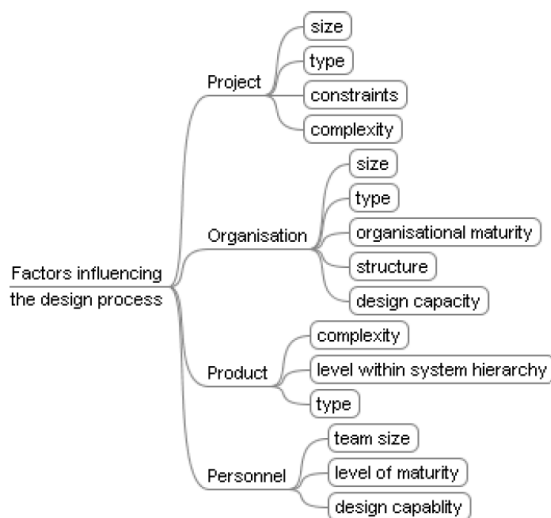


Figure 5.a. Context variables influencing the development process [Ma98]

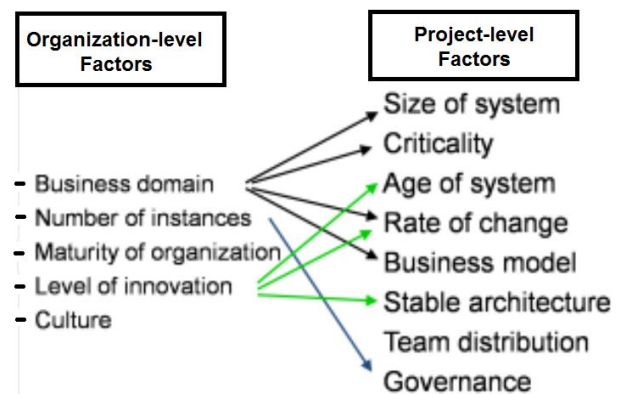


Figure 5.b. Relationships between the 2 sets of factors [Kru09]

Kruchten identifies two sets of factors that make up the context: the *organizational-level factors* and the *project-level factors* (Figure 5.b.) [Kru09]. Typically organization-level factors influence the project-level factors, which in turn should drive the process and practices used in each project. Kruchten also mentions that “in small organizations, with few software development projects, this distinction does not apply and all factors are on the same level”.

Other researchers have also identified a quite similar set of context dimensions/variables. For example Boehm and Turner have identified 5 factors to contrast software process/methods: *size*, *criticality*, *personnel* (particularly skills, know-how), *dynamism* (rate of change) and *culture* (of the team) [BT03]. Cockburn have identified that the *project size*, its *criticality* and *team skills* are drivers for selecting particular processes from the Crystal family of processes [Coc01]. Ambler has identified eight scaling factors (context variables) that must be considered when define, select or adapt a software process. These factors are the following ones: *team size*, *geographical distribution*, *regulatory compliance*, *domain complexity*, *organizational distribution*, *technical complexity*, *organizational complexity*, and *enterprise discipline*. Similar to the previous cases, they need a more detailed definition to be used for tailoring software processes. Moreover, these context attributes were not defined thinking in SME, therefore their suitability for that work scenario must be evaluated.

4 A Domain Specific Language to define Project Context Models

This section presents the definition and implementation of a DSL for software project context modeling. The proposal is explained in detail in the next subsections.

4.1 Language Definition

We have developed a canonical specification for introducing the Software Project Context Modeling Language (SPCML). The SPCML proposal has been obtained from empirical experience of the authors and consultants in software process improvement and definition. In order to do that we have considered the relevant factors for adapting the software process in the practice in SME – Small and Medium Enterprises. Next we present the structure and main components of the proposed language.

4.1.1. Language basic components

The language is composed of two basic components: *context attributes* and *links* between these attributes. The meaning of each context attribute (i.e. its definition) is available in the Annex A. The attributes considered in SPCML were grouped in four context dimensions: project, team, product and process. Next we briefly describe such dimensions, the context attributes involved in such categories, and the set of values that can be assigned to the attributes.

Software Development Project Context CanonicalCase

Dimension Project

ProjectType:	{newDevelopment, extension, maintenance}
Duration:	{short, medium, large}
ClientInvolvement:	{high, medium, low, known}
Problem Knowledge:	{clear, ambiguous, unclear}

TimeConstraints: {veryConstrained, typical, unconstrained}
BudgetConstraints: {veryConstrained, typical, unconstrained}

Dimension Team

TeamSize: {veryRestricted, typical, unrestricted}
TeamExpertise: {high, regular, low}
BussinesKnowledge: {know, affordable, unknown}
ProductKnowledge: {know, affordable, unknown}

Dimension Product

TechnicalComplexity: {high, medium, low}
QualityRelevance: {high, regular, minimum}

Dimension Process

ProcessFocus: {finalProduct, everyProduct}

These dimensions and their attributes have been identified based on the literature review, the experience of the authors and participants in the ADAPTE project. Moreover, in this definition was also considered the particularities of the development scenario that is present in a typical SME. Although in Section 3.4 we have shown a long list of variables (or context attributes) that can be used as part of a project context, just some of them (those included in SPCML) are relevant to tailor or select process models in a SME scenario.

Concerning the *links* considered in SPCML, all of them establish a relationship between two context attributes. The language defines three types of link: *priority link*, *peer-to-peer link* and *no relationship*. A priority link is represented with an arrow that goes from a source attribute to a destination one. In that case any decision made for tailoring or selecting a software process must assign priority to the source attribute over the destination attribute. In other words, the priority link establishes a hierarchical relevance between two context attributes.

The peer-to-peer links indicate that two context attributes are related, but there is no hierarchical relationship between them. Finally, when there is no link between two attributes it means they are independent and therefore can be considered separately to conduct a selection or tailoring process. This information about the links of a project context model eases the decision making during a tailoring process.

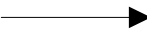

Finally the language considers the configurations, also known as *Project Specific Contexts*, which are particular instances of the project context model for specific development projects. A *Project Specific Context* is a collection of attributes, which are set to one of the possible values indicated by the Context Model. An example of a Project Specific Context for a small project context is specified below. Such configuration, named *small project*, refers to a short and typical extension project. As we can see in the example, the configuration just includes the context attributes (with the respective value) that are relevant for that project context.

ProjectSpecificContext *SmallProject*
{ProjectType = extension, Duration = short, TimeSizeRestrictions = typical}

4.1.2. SPCML graphical representation

Table 3 presents the graphical representation of the elements considered in SPCML.

Table 3. Graphical elements of SPCML

Graphical Element	Meaning
<div style="border: 1px solid black; padding: 5px; width: fit-content;"> <div style="border: 1px solid black; padding: 2px; margin-bottom: 2px;">ContextAttributeName</div> <div style="border: 1px solid black; padding: 2px;">ContextAttributeValue</div> </div>	<i>Context attribute.</i> Each context attribute has a name and a current value which must be set when a configuration is defined. If a context attribute is set as “null”, it means that such attribute does not have to be considered in such configuration.
	<i>Priority link.</i> The link indicates that the two context attributes involved in the relationship must be considered together. In case of priority links, any decision (or rule) related to the source attribute has priority over the decisions (or rules) made for the target attribute.
	<i>Peer-to-peer link.</i> This link is similar to the priority link, however in this case there is not priority to apply the decisions (or rules) defined for the involved attributes.

4.1.3. SPCML conceptual structure

Figure 6 shows the conceptual structure of SPCML, which involves three key concepts: *ContextAttribute*, *Dimension* and *ProjectSpecificAttribute*. As was previously mentioned, a *ContextAttribute* represents a relevant characteristic of the project context that may be required for tailoring. An example of a *Context Attribute* is the *Project Type*. A *ContextAttribute* can take one of a set of values defined as *ContextAttributeValue*. A *Context Attribute Value* represents a specific value for qualifying a *Context Attribute*. Examples of *Context Attribute Values* for the *Project Type Context Attribute* include *newDevelopment*, *extension* and *maintenance*.

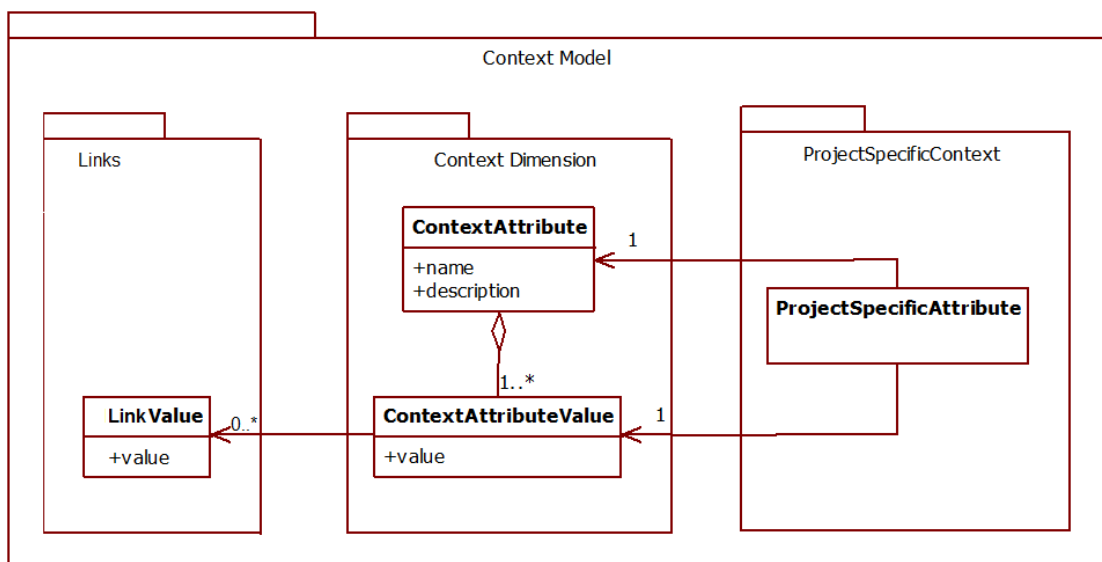


Figure 6. Basic concepts used in SPCML

A *Context Dimension* represents a collection of related *Context Attributes*. A *Dimension* eases the separation of concerns applied to *Context Attributes*. An example of *Dimension* is *Team Dimension*, referring to team attributes such as *TeamSizeRestrictions*, *TeamExpertise BussinesKnowledge* and *Product Knowledge*. A *Context* is represented as a collection of *Dimensions*. A *Context* represents the whole context model. In order to represent possible specific project contexts (also known as configurations), a *Project Specific Context* has be included into the language. A *Project Specific Attribute* is linked to a *Context Attribute*, which is also linked to a unique *Context Attribute Value*. Next section shows the implementation of the SPCML specification.

4.2 Language Implementation

SPCML has been implemented as a meta-model, since the context information manipulation must be done via model transformations, as proposed in the ADAPTE project. Particularly ATL – Atlas Transformation Language– [ATL05] was used as inputs, and the outputs models were defined in XMI (XML Metadata Interchange) format.

The meta-model has been implemented in Ecore, the meta-meta model of Eclipse Modeling Framework, which is presented in Figure 7. This figure shows the above defined context elements, but in this case the relationships between the elements have been detailed.

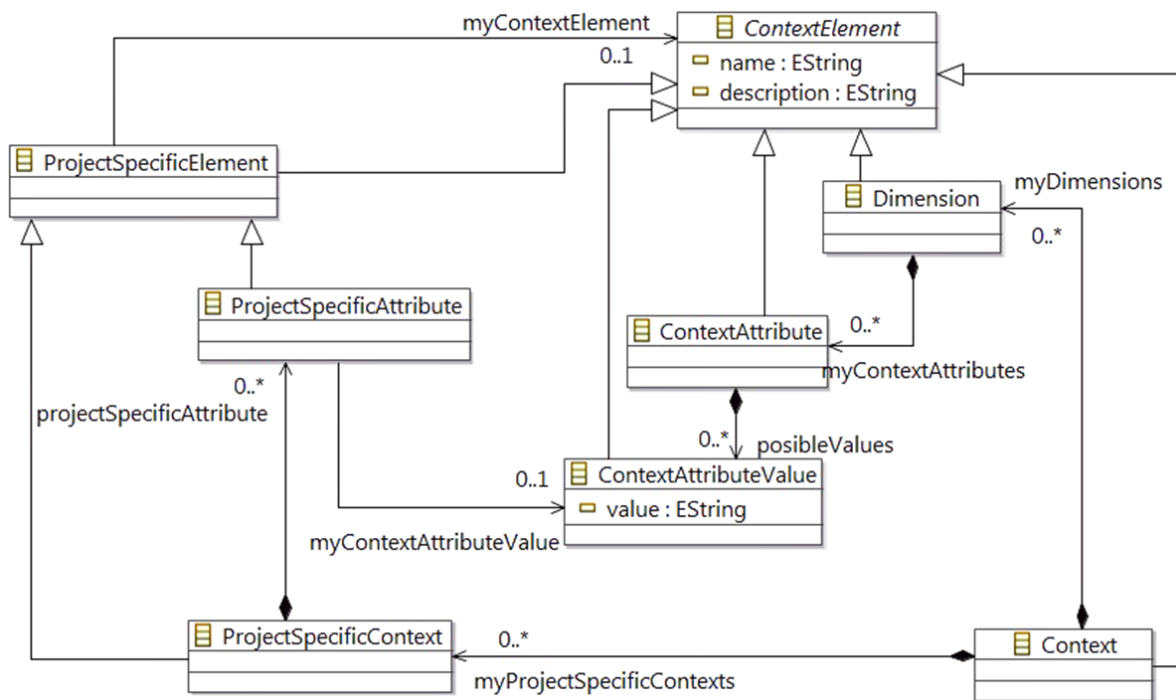


Figure 7. SPCML Meta-model

A canonical definition of the Project Context Model can be specified as a tree using SPCML (Figure 8) and also using a XMI representation (Annex B).

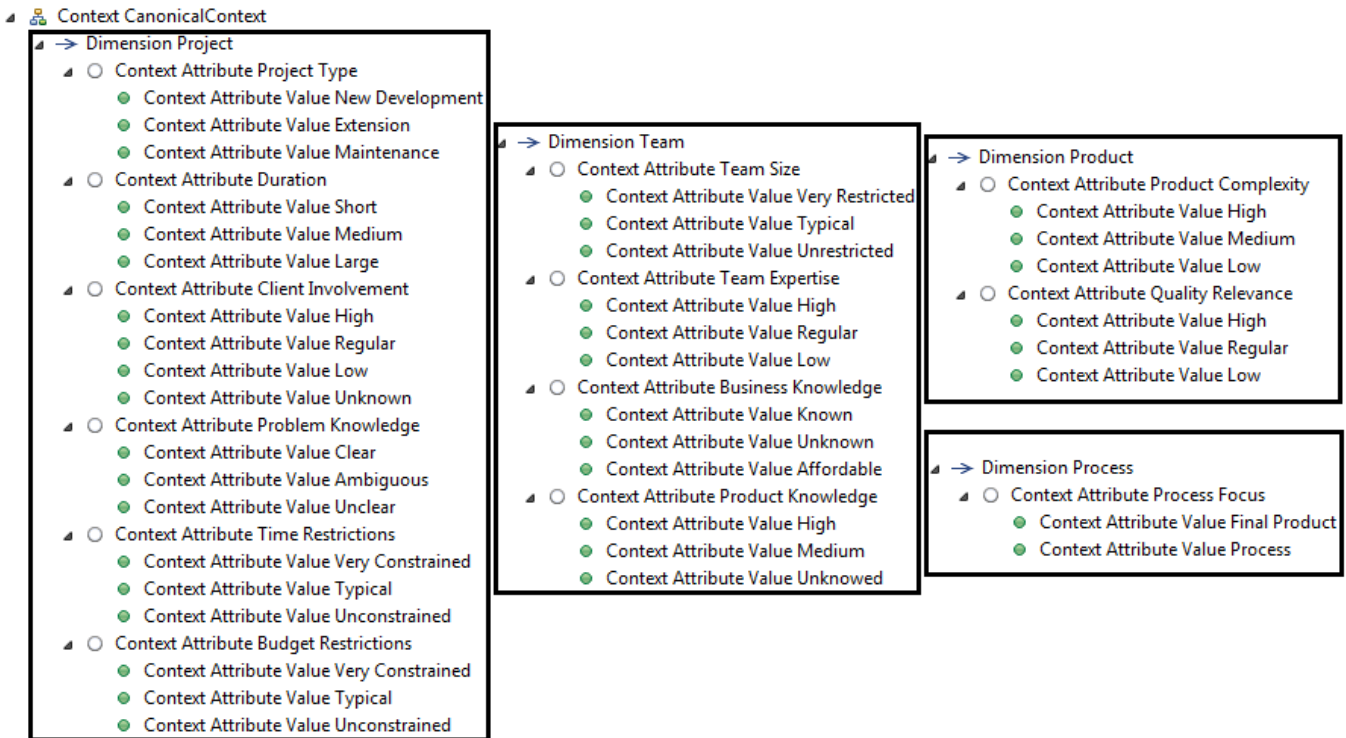


Figure 8. The canonical example represented as a tree

5 Conclusions and Further Work

Dorr et al. [DEE08] suggest that the right set of practices for a project can be better found if we understand the context of the company. Several other researchers such as Stetter [Ste00], Maffin [Ma08] Kruchten [Kru09], Cockburn [Coc01] and Ambler [Amb09] states that the organizational software process have to be adapted to suit the boundary conditions (context) of the specific development project.

In this document we have started reporting the revision and classification of proposals reported by the literature in order to establish a state of the art on the use of context information for engineering software processes. In order to do that we have defined the evaluation criteria to establish the same perspective for the proposals comparison. The results of the literature review indicate that the most frequent use of context information is for tailoring software processes. Particularly, the information is used to determine which process is the most appropriate to guide a certain software development project; i.e. it is for selecting a software process. Most context representations involve textual information or various other types of informal specifications. Just few proposals present a more formal specification, e.g. a Domain Specific Language, which allows us to automatically derive a software process model from other process model.

Provided that in model driven engineering the information is typically specified through models, we have identified the need to define a meta-model to express the project context information. The literature review reported in this work allows us to identify some relevant constructs to define rich context models with a high expressiveness level. Based on these constructs and considering the work scenario that is present in most SME, we have proposed an expressive language named SPCML (Software Project Context Modeling Language) to specify project context. Thus we try to facilitate the application of MDE

strategies in each stage of software process engineering. In this language definition we have used the authors experience and some process engineers involved in the ADAPTE project. The language definition also includes the visual representation of its components and also its implementation in a computable language (i.e. XMI).

A canonical context model for a SME has been defined. The next step is to apply it to some of the SME that are participating in the ADAPTE project. Moreover, a usable graphical interface will be developed for hiding the complexity of the meta-model and the markup language. It will facilitate the context definition to both, process engineers and software engineers.

6 Acknowledgements

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

7 References

- [AKMMNO9] Armbrust, O., Katahira, M., Miyamoto, Y., Münch, J., Nakao, H. & Ocampo, A. Scoping software process lines. *Software Process: Improvement and Practice*, 14, 181-197. 2009.
- [ACG10] Antunes, B., Correia, F., Gomes, P. Context Capture in Software Development. Proc. of the 3rd Artificial Intelligence Techniques in Software Engineering Workshop. Larnaca, Cyprus. October 7, 2010.
- [Amb09] Ambler, S. The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments. White paper. IBM Rational Software. December 2009.
- [AR08] Aharoni, A. & Reinhartz-Berger, I. A domain engineering approach for situational method engineering. *Proceedings of the International Conference on Software Engineering Advances (In ICSEA '08)*, pp. 455-468. 2008.
- [ASRB04] Araujo, R. M., Santoro, F. M., Rosa, M.G.P., Brézillon, P. Context Models for Managing Collaborative Software Development Knowledge. *Proc. of the International Workshop on Modeling and Retrieval of Context (MRC)*, 61-72, 2004.
- [ATL05] ATLAS group. ATL: Atlas Transformation Language ATL Starter's Guide. Version 0.1. LINA & INRIA. December 2005.
- [BB01] Breton, E., & Bézivin, J. Model driven process engineering. In *Proc. of the Computer Software and Applications Conference*, pages 225-230, 2001.
- [BCHWMS95] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. & Selby, R. *Cost Models for Future Software Life Cycle Processes: COCOMO 2.0*. 1995.
- [BD05] Bradley N.A., & Dunlop, M.D. Toward a multidisciplinary model of context to support context-aware computing. *Human-Computer Interactions* 20 (4), 403-446. December 2005.
- [BK05] Bustard, D. W. & Keenan, F. Strategies for Systems Analysis: Groundwork for Process Tailoring. *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, IEEE Computer Society, 357-362. 2005.
- [BKW06] Bucher, T., Klesse, M. & Winter, R. Contextual method engineering. Working paper, University of St. Gallen. 2006.
- [BP99] Brézillon, P., Pomerol, J.-C. Contextual Knowledge Sharing and Cooperation in Intelligent Assistant Systems. *Le Travail Humain*, PUF, Paris, 62, 3, 223-246. 1999.
- [BR87] Basili, V. R. & Rombach, H. D. Tailoring the software process to project goals and environments. *Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, 345-357. 1987.
- [BT03] Boehm, B. & Turner, R. *Balancing agility and discipline*. Addison-Wesley Professional, 2003.
- [CMKC09] Cusumano, M.A. MacCormack, A., Kemerer, C. F. & Crandall, W. B. Critical Decisions in Software Development: Updating the State of the Practice. *IEEE Software*, 26(5):84-87, 2009.

- [Coc01] Cockburn, A. Agile Software Development. Addison-Wesley, 2001.
- [DEE08] Dörr, J. Adam, S., Eisenbarth M., & Ehresmann M. Implementing Requirements Engineering Processes: Using Cooperative Self-Assessment and Improvement. *IEEE Software*, 25(3):71-77, 2008.
- [FH93] Feiler, P. H. & Humphrey, W. S. Software Process Development and Enactment: Concepts and Definitions. *Proc. of the International Conference of Software Process (ICSP'93)*, IEEE Computer Society, 28-40. 1993.
- [FMZ06] Feng, Y., Mingshu, L. & Zhigang, W. SPEM2XPDL: Towards SPEM Model Enactment, 2006.
- [FWT11] Fernandes, P., Werner, C. & Teixeira, E. An Approach for Feature Modeling of Context-Aware Software Product Line. *Journal of Universal Computer Science*, 17, 5, 807-829. 2011.
- [GWJAR09] Gottschalk, F., Wagemakers, T. A., Jansen-Vullers, M. H., Aalst, W. M. & Rosa, M. Configurable Process Models: Experiences from a Municipality Case Study. *Proceedings of the 21st International Conference on Advanced Information Systems Engineering*, Springer-Verlag, 486-500. 2009.
- [HB09] Hurtado, J. A. & Bastarrica, C. Process Model Tailoring as a Mean for Process Knowledge Reuse. *2nd Workshop on Knowledge Reuse (KREUSE'09)*, 2009.
- [HBQ11] Julio A. Hurtado Alegría, María Cecilia Bastarrica, Alcides Quispe, Sergio F. Ochoa. An MDE Approach to Software Process Tailoring. *Proc. of International Conference on Software and system Processes (ICSSP'11)*, co-located with ICSE 2011. Honolulu, Hawaii, USA. May 21-22, 2011.
- [He02] Henderson-Sellers, B. Process Meta-modeling and Process Construction: Examples Using the OPEN Process Framework (OPF) .*Ann. Software Engineering*, J. C. Baltzer AG, Science Publishers, 14, 341-362. 2002.
- [HJM05] Henriksen, K., Indulska, J., McFadden, T. Modelling Context Information with ORM. *Proc. of the On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*. LNCS vol. 3762, pp. 626-635, Springer Berlin/Heidelberg. 2005.
- [Hum89] Humphrey, W. S. The software process. Addison-Wesley Longman Publishing Co., Inc., 1989.
- [HWBBK05] Hanssen, G. K., Westerheim, H. & Bjørnson, F. O. Bomarius, F. & Komi-Sirviö, S. (Eds.) Tailoring RUP to a Defined Project Type: A Case Study. *Product Focused Software Process Improvement. Proc. of the 6th International Conference (PROFES'05)*, LNCS 3547, 314-327, Springer. Oulu, Finland, June 13-15, 2005.
- [KB10] Koolmanojwong, S. & Boehm, B. The incremental commitment model process patterns for rapid-fielding projects. *Proceedings of the 2010 international conference on New modeling concepts for today's software processes: software process*, Springer-Verlag, 150-162. 2010.
- [KDC10] E. Kornysheva, R. Deneckère, & B. Claudepierre. Contextualization of method components. *International Conference on Research Challenges in Information Science (RCIS)*, Nice, France, May 2010.
- [KL05] Kettunen, P. & Laanti, M. How to steer an embedded software project: tactics for selecting the software process model. *Information and Software Technology*, 47, 587 – 608. 2005.
- [Kru09] Kruchten, P. The Context of Software Development. *Kruchten Engineering Services: Weblog*. July 22, 2009. URL: <http://philippe.kruchten.com/2009/07/22/the-context-of-software-development/>. Last visit: Oct., 2011.
- [KSPBKL08] Kang, D., Song, I.-G., Park, S., Bae, D.-H., Kim, H.-K. & Lee, N. A Case Retrieval Method for Knowledge-Based Software Process Tailoring Using Structural Similarity. *Proc of the 15th Asia-Pacific Software Engineering Conference (APSEC '08)*, 51-58. 2008.
- [KSPGS09] Killisperger, P., Stumptner, M., Peters, G., Grossmann, G. & Stückl, T. Meta Model Based Architecture for Software. *Process Instantiation International Conference on Software Process*, 63-74. 2009.
- [LN04] Laplante, P.A. & Neill, C.J. Opinion: The Demise of the Waterfall Model Is Imminent. *ACM* 10-15, 2004.
- [Lo96] Lobsitz, R. M. A Method for Assembling a Project-Specific Software Process Definition. *Proc. of the 29th Hawaii International Conference on System Sciences Volume 1: Software Technology and Architecture*, IEEE Computer Society, 1996.
- [Ma98] Maffin, D. J. Engineering Design Models: Context, Theory and Practice. *Journal of Engineering Design*, 9, 4, 315-327. 1998.

- [MD07] Mnkandla, E. & Dwolatzky, B. Agile Methodologies Selection Toolbox. Proceedings of the International Conference on Software Engineering Advances, IEEE Computer Society, 2007.
- [MKSPM08] Mirakhorli, M., Khanipour Rad, A., Shams, F., Pazoki, M. & Mirakhorli, A. RDP technique: a practice to customize XP. Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral, ACM, 23-32. 2008.
- [MPDG06] Mendoza, L. E., Pérez, M. A., Díaz-Antón, G. & Grimán, A. Tailoring RUP for LMS Selection: A Case Study. CLEI Electronic Journal, 2006.
- [MR05] Mirbel, I. & Ralye, J. Situational method engineering: Combining assembly-based and roadmap-driven approaches. Vol. 11, pp. 58-78, Springer-Verlag. 2005.
- [MW06] Ma, J. & Wang, Y. A Quantitive Context Model of Software Process Patterns and Its Application Method. Proceedings of the 6th International Conference on Quality Software, IEEE Computer Society, 243-250. 2006.
- [PEM96] Perez, G., El Emam, K. & Madhavji, N. Evaluating the congruence of a software process model in a given environment. Proceedings of the Fourth International Conference on the Software Process (ICSP '96), pp. 49-62. 1996.
- [PNS06] Park, S., Na, H. & Sugumaran, V. A semi-automated filtering technique for software process tailoring using neural network. Expert Systems with Applications, 30, 179-189. 2006.
- [RC99] Rus, I. & Collofello, J. S. A Decision Support System for Software Reliability Engineering Strategy. Selection 23rd International Computer Software and Applications Conference (COMPSAC '99), 27-19 October 1999, Phoenix, AZ, USA, 376-384. 1999.
- [RF09] Rolland, C. & Fujita, H. About Strategies to Engineer Situational Methods. Proceeding of the 2009 conference on New Trends in Software Methodologies, Tools and Techniques: Proc. of the 8th SoMeT'09, IOS Press, 22-38. 2009.
- [RK00] Raffo, D. M. & Kellner, M. I. Empirical analysis in software process simulation modeling. Journal in Systems and Software. Elsevier Science Inc., 53, 31-41. 2000.
- [Roy98] Royce, W. Software Project Management: A Unified Framework. Addison-Wesley, Boston, MA, USA. 1998.
- [SAW94] Schilit, B. N., Adams, N. L., & Want, R. Context-aware computing applications. In IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, 1994.
- [Sch06] Schmidt, D.C. Model-Driven Engineering. IEEE Computer 39, 2, 25-31, 2006.
- [SL04] Strang, T., Linnhoff-Popien, C. A context modeling survey. Proc. UbiComp 1st International Workshop on Advanced Context Modeling, Reasoning and Management, Nottingham, 34-41, 2004.
- [Ste00] Stetter, R. Method Implementation in Integrated Product Development. PhD Thesis, TU München, Germany, 2000.
- [ST06] Strode, D. & Tretiakov, A. An Investigation of the Target Environment for Agile Methods Information Systems Technology and its Applications. Proc. of the 5th International Conference ISTA'2006, May 30-31, 2006, Klagenfurt, Austria, 39-50. 2006.
- [WSW07] Wang, Y.-S., Shi, L. & Wang, F.J. A Process Pattern Language for Agile Methods. Proceedings of the 14th Asia-Pacific Software Engineering Conference, IEEE Computer Society, 374-381. 2007.
- [Xu05] Xu, P. Knowledge Support in Software Process Tailoring. Proc. of the 38th Annual Hawaii International Conference on System Sciences, 2005.
- [ZFH05] Zowghi, D., Firesmith, D.G., Henderson-Sellers, B.: Using the OPEN Process Framework to Produce a Situation-Specific Requirements Engineering Method. In Proceedings of the First International Workshop on Situational Requirements Engineering Processes (SREP'05), Paris France, August 2005.
- [ZLO07] Zimmermann, A., Lorenz, A., Oppermann, R. An Operational Definition of Context. in Kokinov et al. (Eds.): Proc. of CONTEXT 2007, LNAI 4635, pp. 558-571, 2007.

Annex A. Definition of the SPCML components

As was mentioned in section 4.1.1, the SPCML language is composed of two basic components: *context attributes* and *links* between the attributes, and these attributes were grouped in four context dimensions: project, team, product and process. Next sections define these dimensions, the context attributes to be considered in each of them, and the values that can be assumed by those attributes.

A.1. Project context dimension

The project dimension represents the main features characterizing the project itself, which are also useful to tailor or select a software process for such context. This dimension involves the following context variables (or context attributes):

Project type: This feature indicates if the project is a *new development*, an *extension*, or a *maintenance* (i.e. replace/re-implement components).

Duration: This variable indicates the current project duration according to the reference values established by each software organization. The values that can be assigned to this project attribute are the following ones: *short*, *medium*, *large*.

Client involvement: This attribute indicates the involvement level of the client (and eventually also the users) in the development project. This involvement level must be quantified considering useful time assigned by the client (and eventually users) to the project. The values that can be assigned to this attribute are: *high*, *regular*, *low*, *unknown*.

Problem Knowledge: This attribute indicates how clear and delimited is the problem to address in a software project. The values that can be assumed by this attribute are: *clear* (the problem is identified, validated and its scope is delimited), *ambiguous* (the problem seems to be identified, but it must be validated and its scope must be delimited), and *unclear* (the problem to address and its scope must be identified).

Time Restrictions: This attribute indicates how tight is the project schedule considering the resources assigned for its development. The values that can be assigned to this attribute are: *very constrained*, *typical*, *unconstrained*.

Budget Restrictions: This attribute indicates how tight is the project budget, considering the resources required for its development. The values that can be assigned to this attribute are: *very constrained*, *typical*, *unconstrained*.

A.2. Team context dimension

The team dimension tries to characterize the task force assigned to a development project. Next we present the attributes that are part of this dimension.

Team size: This attribute indicates how suitable is the task force assigned to the project. The values that can be assigned to team size are: *very restricted* (i.e. the team is minimal for such project),

typical (there is a small room to perform non-mandatory activities), *unrestricted* (there is room to perform non-mandatory activities).

Team expertise: This variable represents the expertise level of the team, considering to the skills required to run the project. The values that can be assigned to team expertise are: *high* (highly appropriated for the project), *regular* (appropriated for the project), *low* (the team has weaknesses to run the project).

Business knowledge: This variable indicates how much does the team knows about the business niche involved in the project. The values that can be assumed by this variable are the following ones: *known* (i.e. the business aspects are under control), *affordable* (the team knows about the business niche, but it has some uncertainty), *unknown* (the team has to learn about the business niche).

Product knowledge: This attribute indicates how much does the team assigned to the project knows about the software product to be developed/extended. This attribute can be: *high* (if e.g. the assigned team –or part of it- was in charge of developing/extending the existing solution or other very similar to that), *medium* (if e.g. the team –or part of it- have access to people having the know-how of the product to be developed/extended), *unknown* (if the team have to learn about the product features, structure and services).

A.3. Product context dimension

This dimension characterizes the product to be developed/extended/maintained through a project. The attributes involved in this dimension are the following ones:

Technical complexity: This variable indicates the technical complexity involved in the activity of developing/extending/maintaining the product committed in a software project. The values that can be assumed by this variable are the following ones: *high* (the functionality is hard to develop, or the involved technology in the project is unknown for the team or immature), *medium* (there is some technical uncertainty on few components), *low* (the development activity and the technology to be used are under control).

Quality relevance: This attribute indicates the relevance that the product quality has in a project. The values that can be assigned are: *high* (product quality is highly relevant in this project), *regular* (the quality is as relevant as in any other project), *low* (quality is not matter of concerns in this development).

A.4. Process context dimension

This dimension determines the relevance of counting on a formal development process in a certain project. This dimension has just one attribute, which is explained next.

Process focus: According to deliverables that must be given to the client, this attribute determine the focus of the process required for a certain project. The values that can be assigned to this attribute are the following ones: *product final* (if the final product is the only (or almost the only) important deliverable of the project), *process* (if the project involves various important deliverables, such as requirements and design documents, and also the final product).

Annex B. Example of an XMI representation of a software project

Next XMI code identifies project dimensions, attributes belonging to such dimensions and a set of values that can be assigned to the attributes.

```
<?xml version="1.0" encoding="ASCII"?>
- <spcm:Context name="CanonicalContext" xmi:id="_u7FiQBq2EeG8k_85Ehp_Zw" xmlns:spcm="http://contextmetamodel/1.0" xmlns:xmi="http://www.omg.org/XMI"
  xmi:version="2.0">
- <myDimensions name="Project" xmi:id="_3pTPk8q2EeG8k_85Ehp_Zw" description="The project dimension represents the main features characterizing the project itself, which
  are also useful to tailor or select a software process for such context">
- <myContextAttributes name="Project Type" xmi:id="_9C8NwBq2EeG8k_85Ehp_Zw" description="This feature indicates if the project is a new development, an extension, or a
  maintenance">
  <possibleValues name="New Development" xmi:id="_CUME8Bq3EeG8k_85Ehp_Zw" value="New Development"/>
  <possibleValues name="Extension" xmi:id="_Gmhq0Bq3EeG8k_85Ehp_Zw" value="Extension"/>
  <possibleValues name="Maintenance" xmi:id="_Lggqk8q3EeG8k_85Ehp_Zw" value="Maintenance"/>
</myContextAttributes>
- <myContextAttributes name="Duration" xmi:id="_Oqy1QBq3EeG8k_85Ehp_Zw" description="This variable indicates the current project duration according to the reference
  values established by each software organization">
  <possibleValues name="Short" xmi:id="_QcHB4Bq3EeG8k_85Ehp_Zw" value="Short"/>
  <possibleValues name="Medium" xmi:id="_S_NQwBq3EeG8k_85Ehp_Zw" value="Medium"/>
  <possibleValues name="Large" xmi:id="_WSY4wBq3EeG8k_85Ehp_Zw" value="Large"/>
</myContextAttributes>
- <myContextAttributes name="Client Involvement" xmi:id="_Ti0FsCziEeGhEf92-7xT4A" description="This attribute indicates the involvement level of the client (and eventually
  also the users) in the development project. This involvement level must be quantified considering useful time assigned by the client (and eventually users) to the
  project">
  <possibleValues name="High" xmi:id="_gCVhcCziEeGhEf92-7xT4A" value="High"/>
  <possibleValues name="Regular" xmi:id="_iIWywCziEeGhEf92-7xT4A" value="Regular"/>
  <possibleValues name="Low" xmi:id="_j6EoACziEeGhEf92-7xT4A" value="Low"/>
  <possibleValues name="Unknown" xmi:id="_mbpz8CziEeGhEf92-7xT4A" value="Unknown"/>
</myContextAttributes>
- <myContextAttributes name="Problem Knowledge" xmi:id="_b9cGACziEeGhEf92-7xT4A" description="This attribute indicates how clear and delimited is the problem to
  address in a software project">
  <possibleValues name="Clear" xmi:id="_2NbDkCziEeGhEf92-7xT4A" value="Clear"/>
  <possibleValues name="Ambiguous" xmi:id="_4IMHICziEeGhEf92-7xT4A" value="Ambiguous"/>
  <possibleValues name="Unclear" xmi:id="_7t30YCziEeGhEf92-7xT4A" value="Unclear"/>
</myContextAttributes>
- <myContextAttributes name="Time Restrictions" xmi:id="_uxoAYCziEeGhEf92-7xT4A" description="This attribute indicates how tight is the project schedule considering the
  resources assigned for its development">
  <possibleValues name="Very Constrained" xmi:id="_iexSMCzjEeGhEf92-7xT4A" value="Very Constrained"/>
  <possibleValues name="Typical" xmi:id="_IuYBwCzjEeGhEf92-7xT4A" value="Typical"/>
  <possibleValues name="Unconstrained" xmi:id="_h60EICzjEeGhEf92-7xT4A" value="Unconstrained"/>
</myContextAttributes>
- <myContextAttributes name="Budget Restrictions" xmi:id="_tIEzMCzjEeGhEf92-7xT4A" description="This attribute indicates how tight is the project budget considering the
  resources required for its development">
  <possibleValues name="Very Constrained" xmi:id="_Glz0CzkEeGhEf92-7xT4A" value="Very Constrained"/>
  <possibleValues name="Typical" xmi:id="_Glz0SzkeEeGhEf92-7xT4A" value="Typical"/>
  <possibleValues name="Unconstrained" xmi:id="_Glz0izkEeGhEf92-7xT4A" value="Unconstrained"/>
</myContextAttributes>
</myDimensions>
- <myDimensions name="Team" xmi:id="_5U8SABq2EeG8k_85Ehp_Zw" description="The team dimension tries to characterize the task force assigned to a development project">
- <myContextAttributes name="Team Size" xmi:id="_b_ibwBq3EeG8k_85Ehp_Zw" description="This attribute indicates how suitable is the task force assigned to the project">
  <possibleValues name="Very Restricted" xmi:id="_fk60EBq3EeG8k_85Ehp_Zw" value="Very Restricted"/>
  <possibleValues name="Typical" xmi:id="_isKR0Bq3EeG8k_85Ehp_Zw" value="Typical"/>
  <possibleValues name="Unrestricted" xmi:id="_bnZ58CzoEeGhEf92-7xT4A" value="Unrestricted"/>
</myContextAttributes>
- <myContextAttributes name="Team Expertise" xmi:id="_tyUt8Bq3EeG8k_85Ehp_Zw" description="This variable represents the expertise level of the team, considering to the
  skills required to run the project">
  <possibleValues name="High" xmi:id="_xGB6YBq3EeG8k_85Ehp_Zw" value="High"/>
  <possibleValues name="Regular" xmi:id="_30IXIBq3EeG8k_85Ehp_Zw" value="Regular"/>
  <possibleValues name="Low" xmi:id="_51gl8Bq3EeG8k_85Ehp_Zw" value="Low"/>
</myContextAttributes>
- <myContextAttributes name="Business Knowledge" xmi:id="_oM3rMCzoEeGhEf92-7xT4A" description="This variable indicates how much does the team knows about the
  business niche involved in the project">
  <possibleValues name="Known" xmi:id="_80HL8CzoEeGhEf92-7xT4A" value="Known"/>
  <possibleValues name="Unknown" xmi:id="_7bv64CzoEeGhEf92-7xT4A" value="Unknown"/>
  <possibleValues name="Affordable" xmi:id="_BFOr4CzpEeGhEf92-7xT4A" value="Affordable"/>
</myContextAttributes>
- <myContextAttributes name="Product Knowledge" xmi:id="_KdGaMCzpEeGhEf92-7xT4A" description="This attribute indicates how much does the team assigned to the project
  knows about the software product to be developed/extended">
  <possibleValues name="High" xmi:id="_SvzQMCzpEeGhEf92-7xT4A" value="High"/>
  <possibleValues name="Medium" xmi:id="_PwIGACzpEeGhEf92-7xT4A" value="Medium"/>
  <possibleValues name="Unknown" xmi:id="_VlxXwCzpEeGhEf92-7xT4A" value="Unknown"/>
</myContextAttributes>
</myDimensions>
- <myDimensions name="Product" xmi:id="_dZytsCzpEeGhEf92-7xT4A" description="This dimension characterizes the product to be developed/extended/maintained through a
  project">
- <myContextAttributes name="Product Complexity" xmi:id="_wmx8f8CzpEeGhEf92-7xT4A" description="This variable indicates the technical complexity involved in the activity
  of developing/extending/maintaining the product committed in a software project">
  <possibleValues name="High" xmi:id="_wmyHACzpEeGhEf92-7xT4A" value="High"/>
  <possibleValues name="Medium" xmi:id="_wmyHASzpEeGhEf92-7xT4A" value="Medium"/>
  <possibleValues name="Low" xmi:id="_wmyHAIzpEeGhEf92-7xT4A" value="Low"/>
</myContextAttributes>
- <myContextAttributes name="Quality Relevance" xmi:id="_9YT4wCzpEeGhEf92-7xT4A" description="This attribute indicates the relevance that the product quality has in a
  project">
  <possibleValues name="High" xmi:id="_EJ7WgCzqEeGhEf92-7xT4A" value="High"/>
  <possibleValues name="Regular" xmi:id="_EJ79kCzqEeGhEf92-7xT4A" value="Regular"/>
  <possibleValues name="Low" xmi:id="_EJ79kSzqEeGhEf92-7xT4A" value="Low"/>
</myContextAttributes>
</myDimensions>
- <myDimensions name="Process" xmi:id="_H7Y18CzqEeGhEf92-7xT4A" description="This dimension determines the relevance of counting on a formal development process in a
  certain project">
- <myContextAttributes name="Process Focus" xmi:id="_N4GsACzqEeGhEf92-7xT4A" description="According to deliverables that must be given to the client, this attribute
  determine the focus of the process required for a certain project">
  <possibleValues name="Final Product" xmi:id="_JKUOCzqEeGhEf92-7xT4A" value="Final Product"/>
  <possibleValues name="Process" xmi:id="_sgnGYCzqEeGhEf92-7xT4A" value="Process"/>
</myContextAttributes>
</myDimensions>
</spcm:Context>
```