A FLEXIBLE PERSPECTIVE FOR SOFTWARE PROCESSES
Supporting Flexibility in the Software Process Engineering Metamodel

Ricardo Martinho
School of Technology and Management, Polytechnic Institute of Leiria, Leiria, Portugal

Dulce Domingos
Department of Informatics, Faculty of Sciences, University of Lisboa, Lisboa, Portugal

João Varajão
Engineering Department, University of Trás-os-Montes e Alto Douro, Vila Real, Portugal

Keywords: Flexibility, software process modeling, process-centered software engineering environments.

Abstract: The lack of flexibility in software process modeling is an important drawback pointed out as the main cause for the low adoption of Process-centered Software Engineering Environments (PSEEs). The Object Management Group (OMG) has been working on the Software Process Engineering Meta-model (SPEM) in order to provide a uniform object-oriented meta-model for building software process models, like the Rational Unified Process (RUP). Nevertheless, the SPEM neither takes into account flexibility aspects nor provides a flexibility meta-model for derived software process models. This paper proposes a flexibility meta-model for building flexible SPEM-based software process models. SPEM compliant PSEEs that implement the proposed meta-model will provide the ability to build flexible software process models, and to associate distinct flexible mechanisms to their corresponding modeling elements.

1 INTRODUCTION

Interest in software process modeling and programming was triggered mainly by Osterweil’s keynote addressed at ICSE 9 in 1987, and his well known quote “Software processes are software too” (Osterweil, 1987). Software processes define sets of activities and related deliverables, data and organizational resources necessary to the production of a software product.

Over the past two decades, advances in process automation technologies and tools have given relevant contributions for the emergence of Process-centered Software Engineering Environments (PSEEs). PSEEs main goal is to manage software process models and related instances. Software process models represent abstract architectures, designs and definitions of software processes that may be instantiated and parameterized in the occurrence of a specific software project. In this context, the Object Management Group (OMG) has been developing the Software Process Engineering Metamodel specification (SPEM), being its latest version 1.1 dated from January, 2005 (OMG, 2005). SPEM is concerned with defining the minimal set of process modeling elements necessary to describe any software development process.

Nevertheless, flexibility is a cross-model property for software process models. Software process flexibility is concerned with the ability of changing or evolving software process models and corresponding instances. For example, a process modeler may want to change a software process model in order to accommodate new activities into the process. The SPEM specification does not provide support for defining flexibility properties for software process modeling elements.

In this paper we propose a flexibility meta-model for SPEM compliant software process models. This meta-model takes into account distinct types of flexibility and related mechanisms, as well as the way they apply for each different SPEM process modeling element.

Below we proceed as follows: section 2 briefly describes the SPEM meta-model. Section 3 provides a classification for flexibility concepts, along with four examples of flexibility requirements. In section 4 we
present our flexibility meta-model for SPEM software processes, based on the classification proposed in section 3. Section 5 discusses related work and section 6 concludes the paper.

2 THE SPEM META-MODEL

To provide the necessary context for the rest of this paper, we resume in this section adapted descriptions according to the last (v1.1) SPEM specification (OMG, 2005).

The SPEM meta-model is used to describe a concrete software development process or a family of software development processes, like Unified Process (UP) based processes (Jacobson et al., 1999). The SPEM meta-model is materialized in a SPEM_Extensions package, that includes five specific subpackages with required modeling elements and semantics for software process engineering, namely:

- **BasicElements** - includes Guidance, GuidanceKind and ExternalDescription elements. A ModelElement is associated with one or more ExternalDescriptions. Guidance elements may be associated with ModelElements in order to provide associated GuidanceKinds that may represent guidelines, techniques, metrics, examples, UML Profiles, tool mentors, checklists or even templates that can be used from a practitioner point of view;
- **Dependencies** - classifies association types between modeling elements, and includes the Trace, RefersTo, Precedence, Impacts, Categorizes and Import dependency types;
- **ProcessStructure** - defines the main structural elements from which a process description is constructed, including the WorkProduct, Activity, Step and ProcessRole modeling elements;
- **ProcessComponents** - includes elements that are concerned with dividing one or more process descriptions into self-contained parts that can be placed under configuration management and version control. Examples include Package, Process and Discipline elements;
- **ProcessLifecycle** - introduces process definition elements to help define the dynamic perspective of the process, that is, how will the process perform over time and its Lifecycle structure in terms of Phases and Iterations.

In the next section we identify and classify flexibility concepts of software processes, in order to extend SPEM to accommodate flexibility.

3 FLEXIBILITY IN SOFTWARE PROCESSES

Software process models have peculiar aspects because they include people that perform creative activities. Therefore, it is difficult to foresee all the lifecycle of a process. Quite often, software projects begin without previous knowledge of all activities, mainly because software projects involve uncertainty. In the next sections we provide examples of flexibility requirements related to a SPEM compliant software process model, and then we identify generic concepts and mechanisms associated with process flexibility and how they can be related specifically with the SPEM approach.

3.1 Software Process Model Flexibility: An Example

To elicit our problem, let us consider the example where a process modeler wants to create a customized SPEM software process model based on UP. The flexibility requirements for the customized process model can be resumed as follows:

- **R1** - Regarding the UP Requirements Discipline, the capture of software requirements may be accomplished by using one of two techniques (Guidances): 1) UML use-case modeling; and 2) prototyping;
- **R2** - The Construction Phase should not be strictly modeled into the process model. Programmers will be responsible for adjusting, during execution time, the corresponding WorkDefinitions;
- **R3** - The software project manager should be able to adapt software quality requirements to conform with new unforeseen directives. This adaptation should affect all running projects, as well as new projects derived from the UP-based process model;
- **R4** - For each UP-based process model Phase, the number of Iterations will depend on the size of the software project, and on the amount of requirement changes occurred during the execution of the project.

Given these requirements, the software process modeler will produce a flexible UP-based process model that can be instantiated for each project that the company gets. In the next subsection we classify flexibility concepts to serve as foundations for the flexibility meta-model proposed in section 4.
3.2 Flexibility Types and Mechanisms

Several relevant works have contributed to solve the lack of flexibility, either for generic Process Aware Information Systems (PAIS) (e.g. (Casati et al., 1998; Sadiq et al., 2005), or particularly for PSEEs (Bandinelli et al., 1993; Heimann et al., 1996; Arbaouei et al., 2002). Based on these contributions, we classify process flexibility in the following categories:

- **flexibility by selection** - the process supports expected exceptions, i.e., it supports alternative paths to normal behavior. These paths may either be modeled: 1) in advance (advance modeling) into the process definition, before execution time, or; 2) as "black boxes", and be specified not before its actual execution time (late modeling);

- **flexibility by adaption** - the process supports unexpected exceptions, i.e., adjustments to process definitions and/or instances to conform to unforeseen, emergent events. Adaptations made at the process definition level are called evolutionary changes and the ones made at the instance level are called ad-hoc changes.

Further on, these categories may be achieved through different mechanisms. For example, in flexibility by selection, advance modeling may be achieved either through prescriptive modeling, where process definitions are rigorously detailed, or through descriptive modeling, where process definitions are reduced to a minimum set of elements, allowing for them to combine in different ways. R₁ is an example of an advanced prescriptive modeling requirement, and R₂ provides a late modeling flexibility requirement.

In flexibility by adaption, there are several alternatives regarding the propagation of evolutionary changes, including (Casati et al., 1998): 1) stopping all old process instances; 2) waiting for all old process instances to complete before starting new ones; or 3) progressively migrate old process instances to comply with new ones. The R₃ requirement is an example of an evolutionary change that should be propagated to all running instances. Ad-hoc changes made to selected process instances may also be enforced using: 1) a definition of a set of operations to change running process instances; 2) defining a new version for the process definition to accommodate the unforeseen changes, and migrating all affected process instances to the new version; or 3) tolerating inconsistencies of process instances with respect to their definitions. R₄ is a typical ad-hoc type of flexibility requirement.

In the next section we present our meta-model that reflects these flexibility concepts applied to SPEM modeling elements.

4 THE SPEM FLEXIBILITY META-MODEL

SPEM modeling elements neither have to be exclusively of one flexibility type, nor one type is implemented in the same manner for each modeling element. For example, a software process Activity may be both flexible by selection and by adaption, being associated with both mechanisms for late modeling and ad-hoc changes. Thus, on one side we have modeling elements that may implement several flexible interfaces, and on the other side, operations defined by those interfaces are implemented according to a particular modeling element.

In Figure 1 we present our proposed SPEM flexibility meta-model. The model implements the facet design pattern, proposed in (Crahen and Ramamurthy, 2002). The facet pattern allows for a clean separation of behavior from its implementation. In this case, the pattern helps us to separate what types of flexibility should a SPEM ModelElement implement, from how can they be implemented, regarding its type (e.g. if it is of type Activity or Phase). Elements in Figure 1 include:

- **FlexibleElement** - represents a flexible element interface. Specializations include sub-types of flexible interfaces that can be implemented by a SPEM ModelElement (the dashed generalization association). Each specialization of FlexibleElement represents a distinct flexible interface, which defines distinct operations for fetching the right FlexibleFacet;

- **ModelElement** - represents the topmost generic modeling element of the SPEM meta-model. ModelElements may implement none to several FlexibleElement interfaces, having, for each one of them, the corresponding operations and the associated FlexibleFacets;

- **FlexibleFacet** - corresponds to concrete implementations of flexible mechanisms, driven by flexibility interfaces implemented by ModelElements. The shared aggregation association means that none to several flex-
ible mechanisms are strictly bounded to a ModelElement’s type.

A process modeler can build her own process model by composing SPEM ModelElements and associating them with the required flexibility mechanisms (implemented through FlexibleFacets). For example, to model the R2 requirement, the process modeler can start by defining the four Phases of the Lifecycle of the process model, and choose the LateModelingFacet for the Construction Phase from a list of available facets. This means that a software project derived from the customized process model could be instantiated and started without the software project derived from the customized process model. Preliminary work starts by considering FlexibleFacets associated with SPEM ModelElements that not only provide flexibility into process enactments but also to process modeling.

REFERENCES


5 RELATED WORK

Important early contributions on software process evolution include the work in (Bandinelli et al., 1993) and the proposed SPADE system. SPADE is a PSEE that supports software process evolution through the reflective features of its custom Process Modeling Language (PML). Process models may be modified using predefined evolution mechanisms offered by the PSEE and tightly coupled to its PML. In (Cugola, 1998) the author proposes the PROSYT PSEE and follows a different approach, enhancing the support of small deviations between real processes and modeled processes. However, none of these works mentioned above promotes a clean separation of types of flexibility, or allows for fine-grained and loose coupled associations between flexibility types and flexible mechanisms for each distinct element of a process model.

In an interesting work presented in (Balust and Franch, 2001), the authors use a similar approach by adopting a UML software process meta-model, and extending it by using a PML called PROMENADE. The definition of flexible software process models is, however, limited to advance and late modeling mechanisms, and no extensible flexibility meta-model is proposed.

6 CONCLUSIONS

This paper proposes a flexibility meta-model for SPEM-based software process models. Main advantages of our approach include: 1) considering SPEM as the reference software process meta-model, as it enhances process modeling standardization and acceptance; 2) using the facet design pattern to enhance a clean separation between flexibility interfaces and corresponding implementations (facets) that depend on each ModelElement specialization; and 3) providing an extensible interface hierarchy, fomenting flexibility type modularity.

We are implementing this flexibility meta-model in a SPEM-compliant PSEE. We are also applying the facet pattern to support distinct flexible perspectives that a process ModelElement can have in our PSEE. Preliminary work starts by considering FlexibleFacets associated with SPEM ModelElements that not only provide flexibility into process enactments but also to process modeling.