A Domain Ontology for Software Process Architecture Description

Fadila Aoussat[1], Mourad Oussalah[1] and Mohamed Ahmed-Nacer[2]
[1]*LINA Laboratoy, University of Nantes, CNRS UMR 6241, 2, Rue de la Houssinière, BP 92208, 44322, Nantes, France*
[2]*LSI Laboratory, Sciences and Technology Houari Boumediene University, BP 32, Bab Ezzouar, Algeria*

Keywords: Software Process Reusing, Software Architectures Principles, Domain Ontology, Software and Systems Process Engineering Metamodel (SPEM), Heterogeneous Knowledge, Ontology Instantiation.

Abstract: This paper presents a part of an approach for software processes reuse based on software architectures. This solution is proposed after the study of existing work on software process reuse field. Our study focuses on approaches for reusing based on software architectures and domain ontology.
AoSP (Architecture oriented Software Process) approach exploits the progress of two research fields that promote reusing for the Software process reusing: Ontology and software architectures.
This article details how the software process architectures are described and discusses the software process ontology conceptualization and instantiation.

1 INTRODUCTION

The quality of the software product depends on the quality of the software process models that is used for the development and the maintenance of this software product. Software Process (SP) models are complex structures used to define the steps performed during the software development. Many kinds of information must be integrated to describe these steps (resources, roles, input and output products...). Therefore, an important number of concepts, paradigms and languages are developed to cover the different development aspects. However, there are always difficulties to model SPs that deal with the software development preoccupations such as understandability, flexibility and dynamicity.

Reuse SPs, is one of the practices used to improve SPs. The objective is to exploit best practices and know-how capitalized from the precedent SP modeling and execution experiments. However, the diversity and the wide range of SP models make SP model reusing very difficult. A number of studies are being conducted nowadays in order to provide better support regarding SP reuse. Unfortunately, no reusing method has emerged as reference in the SP reusing domain.

In order to suggest a solution to cover engineering ”by” reusing SPs, we focus our researches on SP reuse approaches based on software architectures. We think that the reusability, flexibility and abstraction of software architecture are relevant characteristics that can be used to provide a pertinent reusing approach to model high quality SP models.

Moreover, In order to cover the engineering ”for” reusing SPs, we focus our researches on SP reuse approaches based on domain ontology. Our aim is to share common understanding among Stakeholder by capitalizing the best practices of the SP domain. We think that using a domain ontology can manage not only the heterogeneity of the used concepts, but also the heterogeneity of the used terminology.

This paper presents a part of an approach for reusing SPs: AoSP (Architecture oriented Software Process); this approach focuses on the existing approaches insufficiencies and suggests a pertinent solution to reuse SP models. AoSP exploits software architectures principles to model reusable SPs. It describes and deploys SP architectures. On the other hand, to reuse existing SP models, AoSP exploits a domain ontology to capitalize the pertinent know-how extracted from heterogeneous SP models. Our objective is to:

- **Suggest a Generic Solution:** that can be applied for different kinds of SP models.
- **Increase the SP Quality:** we aim to model SPs that have the essential characteristics such as comprehension, modeling and analyzing facilities, agility and execution control.
- **Increase the SP Reusing:** by exploiting the

Aoussat F., Oussalah M. and Ahmed-Nacer M..
A Domain Ontology for Software Process Architecture Description.
DOI: 10.5220/0003973401280133
In *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2012), pages 128-133
ISBN: 978-989-8565-13-6

precedent SP modeling and enactment experiences.

- **Increase the SP Re-usability:** by modeling reusable SP models and handling SP models complexity.

Our article is organized as follows: section-2- summarizes the insufficiencies of the studied reusing approaches. Section-3- presents AoSP approach and the steps to model reusable SPs. AoSP describes SP architectures, thus, Section-4- provides the adopted semantics to describe SP architectures. styles defined for SP architectures. Section-5- details how our domain SP ontology is designed and generated. To allow describing SP architectures, our ontology must capitalize deferent kinds of knowledge, section-6- details how heterogeneous SP knowledge are capitalized. Section -7- concludes the article and announces the future work.

2 INSUFFICIENCIES OF THE APPROACHES FOR REUSING SOFTWARE PROCESSES

2.1 Insufficiencies of the Reusing Approaches based on Software Architectures

In most reusing approaches based on components (Coulette et al., 2000) (Dai et al., 2008) the central concept is the "Process Component". A SP component in an activity (Works Unit) or an activities sequence. SP Component is explicit in most approaches and can be adapted to be reused except for (Dai et al., 2008) where the SP component is considered as black box components and cannot be modified. In general the SP component interface is the Work Product required or given by the SP component (OMG-SPEM, 2008).

The configuration is used in the approaches based on software architectures (Alloui and Oquendo, 2001) (Choi and Scacchi, 2001), however formal rules that describe the assembling of the SP component are not defined explicitly.

For the connector concept there is no consensus on its interpretation (Aoussat et al., 2011), the idea that emerges is that the connector is a dependency between activities, it can be a precedence link or a delegation link, which often depends on the used PML (Process Modeling Language). Each approach defines its own SP connector vision. We resume the insufficiencies of these approaches as follows:

- Limited Reuse: The reusable elements such as SP Component, SP connector are defined to the internal use.
- Under exploitation of architectural elements: Configuration and assembling constraints are not exploited; architectural styles and explicit reusable connectors are not proposed.
- No General Solution: Every approach deals with a particular problem and uses a particular PML.
- No SP Architecture Deployment: No process deployment is proposed.

2.2 Insufficiencies of the Reusing Approaches based on Domain Ontlogy

Table 1: Approaches for reusing SPs based on domain ontology.

Appoach	objective	Ontology
OnSSPKR Framework (He et al., 2007)	Deal with CMM, CMMI, ISO/IEC15504, ISO9001 models.	Three different ontologies
SPO (Software Process Ontologie)(Liao et al., 2005)	Mapping between CMMI model and the ISO/IEC 15504 model	SP basic concepts
PCE based ontology (Tomohiko et al., 1996)	Generate SP plans	Two ontologies (artifacts and activities)
Approach based descriptive logic (Rilling et al., 2007)	Framework for software maintenance	Concepts that affect the software maintenance
Flexible PML based ontology (Shen and Chen, 2006)	Flexible SP model	Process elements

To suggest a domain ontology one of the first steps is to study the existing ones and consider there extension, fusion, adaptation or reuse.

Many SP modeling approach based on domain ontology are defined (He et al., 2007)(Liao et al., 2005)these approaches use one or many ontologies to represent the SP model. However these solutions are specific and deal with particular SP models and do not suggest a general solution that can applied for a large range of SPs.

Table 1- resumes the objectives and the ontologies structures of the studied approaches.

3 AoSP APPROACH DESCRIPTION

AoSP (Architecture oriented Software Process) approach is an approach that gives a solution to increase the SPs reuse. AoSP covers the engineering for and by reusing:

- **For Reusing:** By capitalizing the SP best practices and know-how extracted from existing SP models.
- **By Reusing:** By describing and deploying the extracted software processes knowledge as software architectures.

According to software architectures specificities, AoSP suggests a particular SP modeling approach: SP modeling is decomposed of two steps:

- **Pre Modeling:** Model the different SP preoccupations separately (structure, interaction and treatment). This step increases SP model comprehension and has a direct impact on SP modeling, analyzing and execution control facility.
- **Final Modeling:** Deploy the SP architecture that can be done with different PMLs specific to different SP kinds. The deployment must be in an automatic way by developing deployment programs. This possibility gives to our approach a generic aspect and increases the modeling facility.

4 SOFTWARE PROCESS ARCHITECTURE DESCRIPTION

Based on existing SP reusing approaches insufficiencies, combining with ADL (Architecture Description Language) approaches, AoSP approach suggests a complete semantic to describe and deploy SP architectures.

Our objective is to describe the SP model as software architecture and exploit the advantages offered by the software architecture domain. The interactions have a central place in the SP model (Alloui and Oquendo, 2001), moreover, the SP is human centered; thus, it is important to manage the different kinds of the SP interactions. Our analysis is oriented to give a solution to handle the different kinds of SP interactions. Defining generic explicit reusable SP connectors that can adapt and facilitate the SP interactions is the adopted solution.

We define our SP connector as an activity (Work Unit) that ”facilitate and control” data and control transmissions between SP activities. SP Connectors do not create new products, but adapt, evaluate and control existing products. The distinction between ”creation” activities (SP components) and ”adaptation and control” Activities (SP connectors) is the basis of the SP architectural concepts interpretation. Table–2 resumes the architectural intrperation for SPs.

Table 2: Adopted semantics for Architecture oriented Software Process (AoSP) approach.

Software Process Concepts.	**SP architectural concepts.**
Activity that **creates** new products.	SP Component
Input or output flow of a creation Activity .	SP Port (given or required)
Activity that **adapts or controles** the flow.	SP Connector
Input or output flow of an adaptation Activity .	SP Connector Role
Precedence link between a creation Activity and an adaptation Activity.	Attachement
Delegation link between two activities (adaptation or creation).	Binding
Process structure.	SP Configuration
Recurrent structure or recurrent execution policy.	SP style

5 SOFTWARE PROCESS DOMAIN ONTOLOGY

To capitalize the SP knowledge we use a domain ontology, in addition, our aim is to offer a tool to will allow the reasoning and the emergence of new solutions. Thus our ontology must:

- Be coherent, not ambiguous and commonly accepted.
- Offer a conceptualization to store and retrieve SP architectures knowledge.
- Manage the heterogeneity of the conceptual of the different SPs: offer a conceptualization that can be exploited for different SP models, without focusing on a particular SP kind.
- Manage the heterogeneity at the instance level: Capitalize knowledge from various SP models can create ambiguities, indeed, even if there is consensus on the used terminology for SP modeling, the developers can use their own vocabulary.
- Restore a comprehensible knowledge: A vocabulary reference that represents the vocabulary of the final user must be defined and stored.

5.1 Software Process Ontology Conceptualization

Heterogeneity on the concept level is handled by exploiting the SPEM conceptualization. SPEM (System and Software Process Engineering Metamodel) is a UML profile adopted by the OMG to describe large range system and software processes (OMG-SPEM, 2008). We adopt SPEM as basic conceptualization as it is a standard metamodel accepted by the community, it regroups all the important concepts used on the SP engineering independently from the kind or a concerned domain. The majority of the SP models are conform to the SPEM, or at least, their metamodels can be mapped with SPEM.

SPEM introduces the reusing based process components through the Method Plugin profile; however, to describe SP architectures, SPEM lacks important architectural concepts. In fact, the lack of "SP Configuration", "SP Style" and "explicit Connector" concepts disallow describing and deploying SP architectures (Aoussat et al., 2011). Having a complete semantic to describe a SP architecture, we had extended Method Plugin profile, for this purpose, we had introduced new stereotypes to describe the architectural elements of the SP architectures (Aoussat et al., 2011).

SPEM profile extension is not the subject of the article, we resumes only the architectural elements that describe SP architectures integrated on SPEM. Thus, two kinds of classes are added:

- **Classes that Describe the SP Architecture:** A SP Configuration is composed from SP Components and SP Connectors. the assembling is done via attachments.
- **Classes that Describe the SP Style:** The SP style is composed of Activity Definitions. As SP component and SP connector are activities, an Activity Definition describes the type of the SP connector and the SP component. In the same manner, Work Product Definition describes the types of SP ports and the types of SP connector roles.

5.2 SPEMOntology Structure

SPEMOntology is the result of successive ATL transformations applyed on SPEM. It is constitute from 56 concepts and an important number of data and object properties. In order to facilitate its understanding, it is important to describe its organization. SPEM is structured into seven packages (OMG-SPEM, 2008). By analyzing the SPEM packages (after the extension), we notice that every SPEM package has its abstract class that regroups the common behavior of the packages classes.

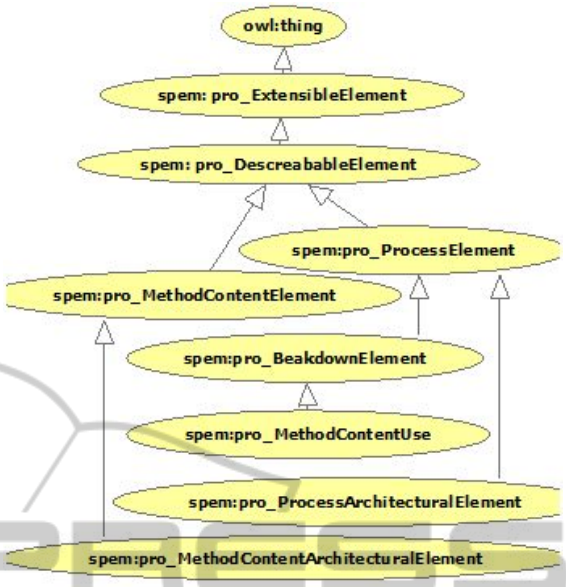

Figure 1: Main abstract SPEMOntology concepts.

After the ATL transformations we can identify this organization (figure -1-). The SPEM packages view can be identified through the main abstract concepts of SPEMOntology. The concepts of our ontology have kept the same name as the SPEM classes; however, we have added the prefix "pro" to identify the stereotyped elements. This prefix is added during the execution of "applySPEMprofile2SPEMmodel" module.

6 SPEMOntology INSTANTIATION

The concepts heterogeneity finds solution by exploiting a standard metamodel. The heterogeneity at instance level deals with separating every kind of knowledge. Indeed, our ontology must store four kinds of knowledge:

- **The SP Architecture Knowledge:** The knowledge concerns SP configuration and SP styles.
- **The used Knowledge:** The knowledge concerns the know-how of existing SP models:
- **The Reference Vocabulary:** The knowledge concerns the vocabulary used by the final stakeholders.
- **The Instance Heterogeneity Management:** our ontology must manage the heterogeneous vocabulary.

Our ontology respects SPEM metamodel concept-

ualization and has the same packages structure. We exploit this structure to deal with the instance heterogeneity. Every SPEM package is used to store a kind of knowledge. We detail the adopted solution in the next paragraphs.

6.1 The SP Architectures Knowledge Capitalization

The SP expert stores the SP configurations and the SP styles of the company. This step is very important as it allows describing formally the company development strategies and practices.

The instantiation is done on the Process Architectural Element concepts that describe the SP configuration behavior and the SP style knowledge is capitalized by using the Method Content architectural concepts (figure-2-).This step is done manually by a SP expert of the company. However, the advantage is that it is done once and it will be reused independently from the SP expert intervention.

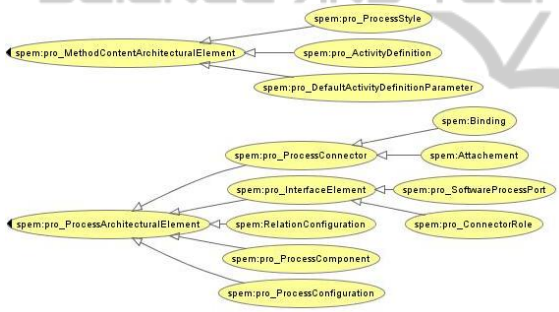

Figure 2: SP architectural concepts of SPEMOntology.

6.2 The used Knowledge Capitalization

We instantiate the concepts of ”Process with Method” and ”Process Structure” packages. We use these concepts to capitalize the used knowhow that are collected from the existing SP models. This step is done automatically; we apply a reverse engineering on every SP model that will be reused. For each PML we develop an instantiation program that identifies the pertinent concepts and allows the extraction of the pertinent knowledge.

6.3 The Reference Vocabulary Capitalization

In SPEM the ”Method Content” package is dedicated to describe development methods independently from their use (OMG-SPEM, 2008). We use these concepts, to describe the vocabulary reference.

The Method content concepts are solicited to describe many kinds of knowledge: Method Content Elements, Vocabulary Reference and Architectural Types. To distinguish between these kinds of knowledge, for each Method Content concept we add a data type property ”concept role” that can have the next values: ”MC” for method content knowledge, ”VR” for vocabulary reference and ”AT” for architectural type.

The weakness of this step is that the instantiation is done manually.

However, the advantage of this manual step is that allow to define ”formally” the glossary of the company. It allows not only a better comprehension of the SP models, but also, constitutes a contribution to capitalize company know-how, that will be used and reused formally independently from the SP experts and its tacit knowledge.

6.4 The Correspondence between the Vocabulary Reference and the used Knowledge

This correspondence is done by using existing associations between Method Content Concepts and Process With Method concepts, these associations are used to define the correspondence between the Used Knowledge and the Reference Vocabulary.

7 CONCLUSIONS

This paper presents a partial view of AoSP (Architecture oriented software Process) approach to reuse SP models. The objective of AoSP is to suggest a standard solution to increase the reuse and the reusability of the SP models.

AoSP offers an innovative vision of the SP modeling by separating the SP modeling preoccupations: Work Product treatments (Components), Work Product transmissions (Data Flow connectors) and execution control (Control Flow connectors). This new vision is possible by exploiting software architectures characteristics; it allows modeling more comprehensible, flexible and controllable SP models. On the other hand, AoSP exploits the precedent good modeling and enactment experiments to model high quality SP models. AoSP uses a domain ontology to capitalize the best practices of the software development domain. It exploits the capitalized knowledge to retrieve and deploy SP architectures.

The ontology conceptualization is discussed, it is based on SPEM; however, SPEM architectural con-

cepts disallow describing SP architectures,thus, we had extended SPEM metamodel by introducing the required architectural concepts. The ontology was generated by transformation model techniques; to achieve this aim, we use ATL (Atlantique Transformation Language).

SPEMOntology must store different kinds of knowledge: The used know-how, the SP architecture knowledge and a reference vocabulary, in addition, it must do a correspondence between these kinds of knowledge. To this aim, we exploit the SPEM structure (organized into packages) to store separately these kinds of knowledge. We add adequate properties to have to keep the knowledge coherence.

Actually we are working on defining inference rules to infer tow kinds of knowledge: "equivalent SP configuration" to identify the SP configurations that can replace the required configuration and "equivalent SP components" to identify the components that can replace the required SP component. We are also working on retrieving SP architectures, the hole algorithm is defined, good results are obtained but must refined before their publishing.

REFERENCES

Alloui, I. and Oquendo, F. (2001). Supporting decentralised software-intensive processes using zeta component-based architecture description language. In *ICEIS*, pages 207–215.

Aoussat, F., Oussalah, M., and Nacer, M. A. (2011). Spem extension with software process architectural concepts. *Computer Software and Applications Conference*, 0:215–223.

Choi, S. J. and Scacchi, W. (2001). Modeling and simulating software acquisition process architectures. *Journal of Systems and Software*, 59(3):343–354.

Coulette, B., Thu, T. D., Crgut, X., and Thuy, D. T. B. (2000). Rhodes, a process component centered software engineering environment. In *ICEIS*, pages 253–260.

Dai, F., Li, T., Zhao, N., Yu, Y., and Huang, B. (2008). Evolution process component composition based on process architecture. In *International Symposium on Intelligent Information Technology Application Workshops*, pages 1097–1100.

He, J., Yan, H., Liu, C., and Jin, M. (2007). A framework of ontology-supported knowledge representation in software process. http://www.atlantis-press.com/php/download_paper.php?id=1180.

Liao, L., Qu, Y., and Leung, H. K. N. (2005). A software process ontology and its application. In *Workshop on Semantic Web Enabled Software Engineering(SWESE)*.

OMG-SPEM (2008). SPEM:Software & Systems Process Engineering Metamodel, v2.0. http://www.omg.org/cgi-bin/doc?Formal/2008-04-01.

Rilling, J., Zhang, Y., Meng, W. J., Witte, R., Haarslev, V., and Charland, P. (2007). A Unified Ontology-Based Process Model for Software Maintenance and Comprehension. In *Models in Software Engineering: Workshops at MoDELS*, volume 4364, pages 56–65.

Shen, B. and Chen, C. (2006). The design of a flexible software process language. In *SPW/ProSim*, pages 186–194.

Tomohiko, K. M., Mori, K., and Shiozawa, T. (1996). Process-centered software engineering environment using process and object ontologies. In *the Second Joint Conference on KnowledgeBased Software Engineering*, pages 226–229.