REUSING APPROACH FOR SOFTWARE PROCESSES BASED ON SOFTWARE ARCHITECTURES

Fadila Aoussat, Mohamed Ahmed Nacer
Department of Computer Science, Saad Dahlab Blida University, BP270, Route Soumaa, Blida, Algeria
Department of Electronique and Computer Science, University of Sciences and Technology Houari Boumediène BP32, ElAlia, BabEzzoua, Algeria

Mourad Oussalah
LINA Laboratoy, University of Nantes, CNRS UMR 6241, 2, Rue de la Houssinière, BP 92208, 44322, Nantes, France

Keywords: Reuse based component, Software processes domain ontology, SPEM metamodel, Software architecture concepts.

Abstract: Capitalizing and reusing the knowledge in the field of software process engineering is the objective of this work. In order to ensure a high quality for software process models, regarding to the specific needs of new development techniques and methods, we propose an approach based on two essential points: The Capitalization of the knowledge through a domain ontology, and the reusing of this knowledge across handling software process models as software architectures.

1 INTRODUCTION

The fast growth of technology and development tools added to the continuous change of development practices and traditions (component-oriented development, pair programming,...), often suggests new methods and new development processes (UP, XP, 2TUP....). Modeling software processes with high-quality requires experience and a confirmed expertise, regarding years of reflexion and refinement. Reusing software processes that have been previously developed, tested, used and that have proven their efficiency is the main objective of our work.

Several approaches for modeling software processes based components have been proposed (Gary,1998),(Avrilionis,1998),(Dami,1998), (Hiltomi,1996),(Thu,2000), (Drai,2008), (Belkhatir, 1996), Most of these approaches use the concept of "Component Software Process" described as a fragment or a part of a software process. However, as reusing components approaches, each approach offers its own solution, addressing a particular aspect of modeling and executing software processes.

The major weakness of these approaches is that the developed software process components are specific to the environment; the use of the software process components is limited to the environment itself. Indeed, these environments operate independently and do not reuse software process components developed in other environments. In the same context of reuse scope, the concept of software process component On The Shelf "ready for use" has not yet appeared; so the immaturity and newness of this area is a logical justification for this work.

Based on the richness of the field in terms of concepts and experiences, as well as the limitation of existing approaches (software processes components weakly reusable, architectural abstraction not taken into account), we propose a new approach that has as main goal to expand and to facilitate a relevant reuse of software process models in term of knowledge. Our solution is based on the use of a domain ontology which capitalizes this knowledge to allow an inference of new software process models. By focusing on the architectural abstraction and addressing the software process as pure software architecture, solutions can be proposed for more efficient reuse. Therefore, the software processes that we develop are software processes based on

Aoussat F., Ahmed Nacer M. and Oussalah M. (2010).
REUSING APPROACH FOR SOFTWARE PROCESSES BASED ON SOFTWARE ARCHITECTURES.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Databases and Information Systems Integration*, pages 366-369
DOI: 10.5220/0002876103660369

Table1: Approach oriented object characteristics.

Component Characteristics	Environment RHODES	Framework OPC	PYNODE	ENDEAVORS	APEL
Creating period	Before the reuse	During the reuse	During the reuse	Adapted during the execution.	Before the reuse
Processes Modeling Language(PML)	PBOOL+ (Object oriented)	Object oriented languages	Object oriented languages	ObjV based OOP LISP	Not specific language.
Heterogeneity	Homogeneous	Syntaxic	Syntaxic	Homogeneous	Syntaxic/Semantic
Assembling	Static	Dynamic and Incremental.	Dynamic and Incremental.	Static	No assembling
Metamodel	SPEM	Use all concepts of the metamodel			
		Basic elements (role/activity/artifact)	Basic elements	Basic elements	Basic elements (activity, resource artifact)
Executing plateformes	Same platform	Same platform	Same platform	Multiple	Multiple
Component identification	Not assisted	Half assisted		Half assisted	No identification.
Reusability scope	Internal to the system				
Configuration management	No management (graphical representation of the assembly)				

software architecture. So, for handling and describing software architectures processes we will inspire from the existing ADL (Architecture Description Language).

The paper is organized as follows: Section 2 summarizes existing approaches for modeling processes based on software components; the objective is to focus on the strengths of these approaches, and particularly, to detect their lacks. Section 3 presents the general outlines of our approach to modeling software process based software architecture. Our approach is based on the use of a domain ontology that contains software process knowledge. Section 4 details the essential points for creating the ontology and discusses the encountered problems and the possible solutions. We conclude the paper summarizing the work with the future research.

2 EXISTING APPROACHES FOR THE REUSE OF SOFTWARE-BASED METHODS OF COMPONENTS

We distinguish tow kind of approaches: approaches of the model level (M1) of the OMG modeling architecture, and approaches of the metamodel level (M2).

2.1 Model Level

Several approaches to software process modeling based components have been developed. Each approach offers a particular solution, focusing on the concerns of its user, as the heterogeneity of languages process modeling software (Gary, 1998) (Avrilionis, 1998), the heterogeneity of execution platforms (Hiltomi, 1996), the distributed execution (Dami, 1998) or the conformity with SPEM meta-model(Thu, 2000). The major weakness of these approaches is that the components are specific and their use is limited to their original environment. These systems typically operate so independently and do not reuse "external" components of their software processes. The studied approaches use object-oriented languages for software process modeling; they implement their components as classes and use the object mechanisms (inheritance, instantiation ...) (Table 1- line -2-).

2.2 Metamodel Level

SPEM (Software and Systems Engineering Metamodel) (OMG-SPEM, 2008) is a metamodel that describes a large range of software processes. Its organization into multiple packages offers not only several view points on the software processes (method view, structure view, reuse view ...), but also, facilitate the expansion and integration of new concepts.

SPEM supports different types of reuse: on one hand, while specifying "Process Behavior" package to capture external behavior of software process models that are not conform to SPEM metamodel, and on the other hand, while introducing reuse based on software process Components by providing another package: “the Method Plugin package”.

However, reusing components in SPEM faces several "recognized" problems that must be treated. The most important are the interconnection problems of components: heterogeneity of the terminology used for the port component "Work

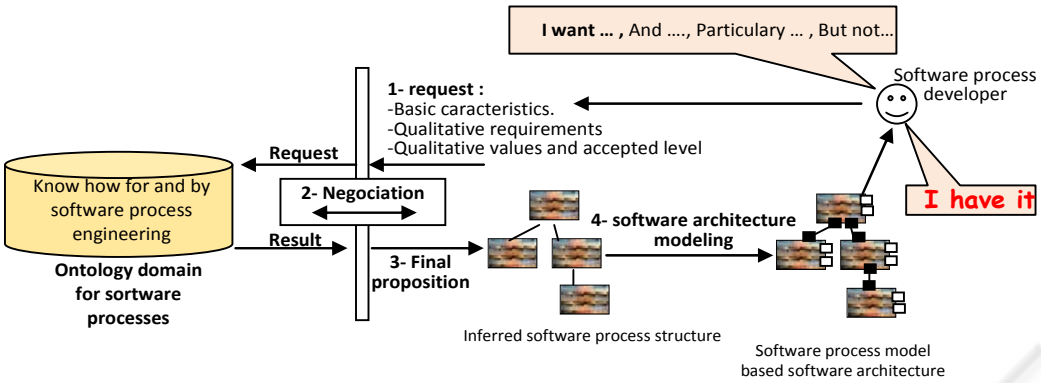

Figure 1: Software process modeling based on software architecture inferring.

Product Port", the management of the number of ports per component creates difficulties for assembling components.

3 OUR APPROACH

The main contribution of our approach is that we model software processes as software architectures (Boehm, 1996). We model the content of software processes regardless of their structure, and we model the logical structure independently from the software process implementation. This separation is one of the characteristics of software architectures; that's allows us greater flexibility during the modeling process management and better control when modeling different kinds of software processes.

3.1 Engineering for Reuse

This step attempts to remedy the low reusability of software process components and to take advantages of the maturity of the field in terms of experiences and conceptualization. To capitalize the knowledge of the domain, the proposed solution is the use of a domain ontology including most concepts in the software processes field. The ontology will form a support that contains the knowledge of this area, which will be reused regardless of their original environment. The instantiated ontology becomes a knowledge base, from which we can infer principally new software process models based on software architecture.

3.2 Engineering by Reuse

Engineering by reuse is occurring by the inference of new software process models from the ontology knowledge. The query must consider the request of the process developer and then infers the knowledge that matches developer requirements.

The query should allow the software process architecture inference, should identify software process components and their configuration (assembly). The assembly can be conform to a software process architectural style as it cannot be.

4 DOMAIN ONTOLOGY FOR THE INFERENCE OF PROCESSES BASED SOFTWARE ARCHITECTURE

To capitalize the knowledge of software process engineering, our solution is based on a domain ontology. To collect the concepts of our ontology, we exploit existing conceptualizations involving the basic concepts for modeling and executing software processes. Our work was oriented to the SPEM metamodel, that is more general, not specific to an environment and includes the concepts of several software process types. To create our ontology, we generate our ontology SPEMOntology "automatically" from the SPEM metamodel, we use the models transformation language ATL (ATL, 2006).

An ATL transformation is composed of ATL modules. For our generation we use three existing transformation modules: UML2OWL OWL2XML and UML2Copy. However, for our work, this transformation is not sufficient as it does not transform a "stereotyped" UML model conforms to a UML profile into an OWL model. The transformation UML2OWL does not contain transformation rules applied to profiles and their constituents (stereotypes, constraints and tagged values). Therefore, we define a new transformation

(ATL1) which applies the profile SPEM model SPEM. Finally, the successive transformation (ATL1, ATL2) generate our SPEMOntology.

5 CONCLUSIONS

In this paper we explore the problem of limited reuse for software processes. We first identified the shortcomings of existing approaches; in fact, many approaches was proposed for modeling software processes based components, focusing however on a particular problem. Also, as the reasoning on the architectural abstraction level is not being a priority; the representation of architectural concepts is insufficient. The classification of these approaches in engineering “for” reuse can justify the absence of some concepts such as the logic configuration; however, it does not justify the low representation of other concepts.

Our paper introduces the general outlines of a new approach to modeling software processes based components. Our approach tempts to remedy the shortcomings of existing approaches (low reusability of software components, architectural concepts poorly exploited) and to exploit the reuse to its extreme: in fact, due to the rigidity and the dependency of software process to their development environment, high quality process models are developed and are not "re" exploited.

We believe that the exploitation of the architectural level of software processes will not only allows the effective reuse of knowledge in software process domain, but also, contributes significantly to facilitate and to resolve the modeling problems, the execution and the simulation of different software process structures:

The validation of our proposition is underwork. Mutiple points remain to be developed: the extension of the ontology and the extension of SPEM with architectural concepts for software processes are the next targets of our work.

REFERENCES

Gary, K., Lindquist, T., Koehnemann, H., Derniame, J.-C. 1998. Component-based software process support. *13th IEEE AS E’98.* Page(s):196- 199.

Avrilionis, D., Belkhatir, N., Cunin, P.-Y, 1996. A unified framework for software process enactment and improvement. *ICSP’96*. Page(s):102 – 111.

Dami, S., Estublier J., Amiour, M. January 1998. APEL: A Graphical Yet Executable Formalism for Process Modeling. *ASE’98*. Vol. 5, No. 1, pp. 61-96.

Hitomi, A. S., Bolcer G. A., Taylor, R. N., 1996. Endeavors: A Process System Infrastructure. *ICSE’96.*

Thu, T. D., Bich Thuy, D. T., Coulette B., Cregut X., 2000. Process component modeling: integration in the RHODES environment. *Journées francophones d'ingénierie des connaissances, Toulouse, France.* pp.165-173.

Object Management Group, Software & Systems Process Engineering Meta Model, v2.0 http:// www.omg.org/cgi-bin/doc?Formal/2008-04-01.

Zamli, K. Z., Mat Isa. N. A., 2004. A survey and analysis of process modeling languages. Malaysian Journal of Computer Science, Vol.17 No. 2 , pp. 68-89

Dai, F., Li, T., Zhao, N., Yu, Y., Huang B., 2008. Evolution Process Component Composition Based on Process Architecture, *ISIITA Workshops.* Volume 00, Pages 1097-1100.

Borsoi, B. T., Becerra J. L. R., 2008. A Method to Define an Object Oriented Software Process Architecture. *ASWEC*’08, *pages: 650-655.*

Boehm, B., 1996. An Open Architecture for Software Process Asset Reuse. *ISPW’96.*

ATLAS group LINA & INRIA ATL: Atlas Transformation Language, 2006. ATL User Manual, version 0.7, Nantes.

Belkhatir, N., Estublier, J., 1996, Supporting Reuse and Configuration for Large Scale Software Process Models. *ISPW’96.*