SPEM Extension with software process architectural concepts

Fadila Aoussat, Mourad Chabane Oussalah, Ahmed-Nacer Mohamed

SPEM Extension with software process architectural concepts

Fadila Aoussat
Computer Science Departement,
Saad Dahlab Blida University,
BP 270, route Soumaa,
Blida, Algeria.
Fadila.aoussat@univ-nantes.fr

Mourad Oussalah
LINA Laboratoy,
University of Nantes,
2, Rue de la Houssiniere,
BP 92208, 44322, Nantes, France.
Mourad.oussalah@univ-nantes.fr

Mohamed Ahmed Nacer
LSI laboratory
USTHB University,
BP 32, El Alia,
BabEzzouar, Algeria
Anacer@mail.cerist.dz

***Abstract*—SPEM is a metamodel adopted by the OMG for software processes engineering. For the software process architectures description, SPEM architectural concepts are very insufficient. Indeed, the existing concepts disallow describing configurations and explicit links specific to software process architectures and finally their deployment.**

The objective of this paper is to present an extension of SPEM (System and Software Process Engineering Metamodel) with lacking architectural concepts. This extension is an important step to implement a new approach for software process reusing based on software architectures.

***Keywords*-Software process architecural concepts, Method Plugin Profil, explicit process connectors, software process styles.**

I. Introduction

The agile methods are defined to treat the traditional development methods insufficiencies, mainly, the rigidity and the difficulty to handle continuous changes. These continuous evolutions concern principally the customer requirements, project planning and development priorities.

The agile methods are described as iterative and incremental, they are carried out in a collaborative spirit implying developers and customer. It generates a high-quality product taking account the the customer requirements changes. However, the agility of these methods can become a disadvantage if it is not controlled, thus, the success of these methods depends on two essential points:

- The process quality which must be very flexible and dynamic, and at the same time, must handle the probable deviations, as the project is often based on a confidence contract between customer and developers than a traditional contract where the customer requirements are preliminary fixed.
- Developer's capacities of communication and interaction as well as their experiment and knowhow concerning the development and project the management.

To increase the flexibility of the used software process models by exploiting the precedent project development experiments, modeling SPs as software architectures by reusing existing SPs models is the proposed solution. We propose a new approach for SP modeling based on software architectures, this approach exploits the SPEM metamodel as basic conceptualization to describe and deploy SP architectures. However, SPEM lacks important architectural concepts to describe SP architectures. In fact, the lack of "Process Configuration", "Process Style" and "explicit Connector" concepts disallow describing and deploying SP architectures.

This paper presents the SPEM metamodel extension by the integration of the lacking architectural concepts for SPs modeling based on software architectures. SPEM being a UML profile, the objective is not to evaluate the capacities of UML metamodel to model architectural concepts, but to extend SPEM for the needs of our approach. The goal is to exploit UML2.0 concepts and mechanisms without focusing on its insufficiencies concerning software architecture concepts modeling.

Our paper is organized as follows: section -2 - presents the SPEM metamodel Insufficiencies, section -3- summarizes our approach for software processes reusing based on software architectures. Section -4- presents the adopted approach to the SPEM extension, than we suggest a generic metamodel for modeling SP architectures, section-5- presents the effective SPEM extension, the identified stereotypes and concepts. We conclude the paper by section -6- that summarizes the carried out and future works.

II. Insufficiencies Software Architecture Concepts in SPEM Metamodel

SPEM is a metamodel (UML profile) adopted by the OMG proposed to define software systems and processes developments, it defines the needed concepts to describe a large range of software processes development without focusing on a particular type [1]. The SPEM metamodel conceptual core is constituted on the basic concepts of every SP (Figure -1-).

The SP is a "Unit Work" sequence. The "Unit Work" requires input products to give output products. As the software processes are human centered, a "Role" is responsible on "Unit Works". The presence of other process concepts as "Resource", "personnel", "guidance"...etc depends on the SP type and orientation (resource oriented, personnel oriented,

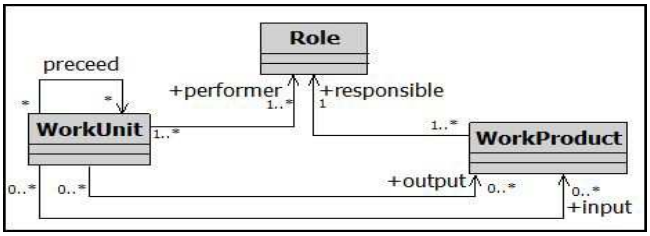

Figure 1. Software processes conceptual core.

guidance oriented...etc). SPEM gives an abstract stereotype ”Extensible Element” in order to specialize these different concepts and to describe different SPs orientation.

SPEM treats SPs reusing based process components through the Method Plugin profile [1]. The Method Plugin profile introduces architectural concepts for reusing based process components through the definition of stereotypes dedicated for this purpose.

According to SPEM a process component ”Process Component” corresponds to exactly one process represented by exactly one Activity ”Activity”. A component has several ”Work Product Ports”, the ”Work Product Port” is an input or an output product, and it is described by a ”Work Product Definition”. A ”Work Product Port” is the port of one and only one ”Process Component”. The Process Components are connected by using ”Work Product Port Connector”. The ”Work Product Port Connector” is used to connect the ”Work Product Ports” of the Process Components.

”Configuration” concept is used but not as architectural structure of SPs. The ”Configuration Method” is not considered as ”Process Element”, but as a ”Classifier” that describes a logical sub set of ”Method Plugins”. A ”Method Plugins” are packages that describe a physical container of ”Process Package” and ”Method Content” package [1]. ”Method Configuration” do not describes the architectural abstraction of the SP model but only the content elements of the SP model.

Important Process Component interconnection problems were identified [1], we notice:

- Difficulties to manage terminology heterogeneity used for the ”Work Product Ports”, as to link Process Components we associate ”Work Product Ports” instances manually by doing a correspondence between their ”different” names [1].
- Difficulties to manage the number of ”Process Component Ports”, particularly when the ”Work Product Ports” number of the connected Process components is different [1].
- The Process components assembly is done from scratch, in ad hoc way, no life cycle or well-known structures are formally exploited, that increase the Stakeholders tacit knowledge dependency [1].
- Manual assembling of the Process Components, that depends on the experiment of the stakeholder [1].

These problems result from lacking of important architectural concepts. By comparing the admitted software architecture concepts with those of SPEM we notice:

- ”Explicit Process Connector” absence, the connector ”Work Product Port Connector” is an implicit connector, it’s a simple direct link between ”Work Product Ports” that assures the precedence or delegation links between Process Components. It hasn’t any role to facilitate or adapt connection between Process Components; no facilitation or adaptation mechanisms are integrated.
- ”Role Connector” concept absence, that explains The direct connection between Work Product Ports.
- According to the cardinalities of SPEM, a connector can connect several ports without any constraints, no constraints assembling are used. In addition, other properties describing the software connector (semantic, evolution, nonfunctional properties) as an architectural concept are not taken into account [2].
- ”Configuration” concept absence: Architectural abstraction of SPs is not supported. ”Method configuration” concept is defined but not specific for reusing based Process Components, it’s a ”Classifier” and not considered as ”Process Element”, it’s a logical description of sub sets of ”Process” and ”Content” package elements without structural vision.
- The topological constraints are not formally exploited for the modeling by reusing SP components. The well-known and recurrent process structures are not exploited for SP modeling based components.

Method Plugin being a UML profile, the divergence in the architectural concepts representation based on UML explains the insufficiencies noted in this profile [1]. Also, few approaches for software process architectures modeling was proposed comparatively with classical software process modeling approaches.

III. Approach For Software Processes Models Reusing Based On Software Architectures

Our solution exploits the progress of two research fields which promote the reusing for the software processes reusing: Ontology and software architectures. It’s constituted on two steps:

1) Knowledge capitalization by reverse engineering applied to existing SPs models. For this main we use domain ontology that capitalizes the pertinent knowledge.
2) The effective knowledge reusing across describing and deploying the extracted software processes knowledge as software architectures.

The purpose of our approach is to offer range choices to model a high quality SPs, by reusing high quality

knowledge. Our preoccupation is, to answer the specific and personalized requests of the stakeholders by exploiting the positive experiments of existing SP models. Also, we aim to solve difficulties of modeling and execution of SPs by treating it at the structural level.

This solution is more dedicated for modeling and executing processes with adaptable structures: dynamic, incremental, iterative, heterogeneous or distributed processes. Handling SP models as software architectures allows greater flexibility for reusing. Separate the process content from the process structure reduces the SP models dependency to their environment and modeling languages.

The paper treats the SPEM metamodel extension with lacking architectural concepts, it presents a solution for encountered problems in the ontology conceptualization. Our ontology respects the SPEM conceptualization, this choice is justified by two reasons: the need to exploit conceptualization accepted by the SPs community, and the large range SP concepts that SPEM covers. It will allow generalizing the approach to processes that are not necessarily oriented development.

However, to describe and deploy SP architectures SPEM lacks important architectural concepts as "explicit connector", "configuration" and "style", so, the SPEM metamodel extension with lacking architectural concepts is needed.

IV. Adopted Method for the SPEM Metamodel Extention

The lacked architectural concepts of SPEM are identified, The SPEM extention is the next step, to this purpose we proceed as follows:

1) Having the SPEM architectural concepts, we collected all the other SP architectural concepts that are used in the existing approaches for modeling and executing SPs based architectures and components [8][11][19][17][20][3][21]. This step has as result the identification and formalization of the SPs architectural concepts of the existing approaches.
2) We explore formal propositions of software architectures description based on UML. We study the proposed metamodels based UML that regroup the architectural concepts, and analyze their conceptualization in order to suggest a generic metamodel based UML specific to SPs architectures. The SP architectural concepts identification is based on ADL (Architecture Description Language) approach [3][17][18][19], as the ADLs have a more pertinent semantic than the traditional architecutres modeling approaches,the ADLs introduce explicitly architectural concepts, techniques and the tools that allow describing software architectures rigorously. This step has as results the introduction of the lacking architectural concepts, the semantics refinement of the existing architectural concepts, then the formalization of our metamodel for SP architectures.
3) Explore the different approaches of UML extension for software architectures [7]. In our work we extend an existing profile, we focus on the profile dedicated to the reusing based process components which is the "Method Plugin profile". However, the difficulty of the extension resides on the manner and the representation choice for modeling the architectural concepts. In fact, according to the existing software architecture descriptions based UML [9][10][11][12][14][16][6], there is no real consensus about the architectural concepts representation. Various notations and approaches are proposed, the "Concept Type", the "Concept" itself, and its "Instances", can be stereotyped using different metaclasses. So, for example, the metaclasses "Class", "Component", "Subsystem", "Package" are used to define stereotypes for the "Component" concept, and the metaclasses "Class", "Connector", "Collaboration", "Association" are used to define stereotypes for the "Connector" concept.

A. Architectural Concepts of the Existing Approaches

To identify the needed architectural concepts for our metamodel, we identify first the existing architectural concepts that were defined in the existing approaches, we note that:

- In the approaches oriented process components the central concept is the "process component". According to the SPs core concepts The "Composite Process Component" is described as software process fragment that represents activities (Unit works) sequence.

Software process concept	Software architectural concept
Process Fragment(Work Units set).	Composite Component.
Elementary Activity (Work Unit).	Elementary Component.
Product(input/output).	Port (required/ given).
Software process structure.	Configuration.

Table I
Mapping concepts of the existing approaches.

- The "Elementary Process Component" is described as an "Elementary activity" (a Unit Work).
- The "Port" (required or given) is represented by a "Work Product" (input or output).
- The "Process configuration" concept represents the abstract structure of the SP model. it's defined in the approaches oriented architectures.
- The "Role" responsible of the "Unit Work" and the other concepts as "tool", "personnel" and "guidance" have no direct correspondence with the architectural concepts, they constitute a part of the SP Component.
- For the "connector" concept there is no consensus on its interpretation, however, the idea that emerges is that the connector is "a dependency between activities", it can be a precedence links or a delegation links. Each

Existing approaches	Connector intrepretation	Explicit
PYNODE [20]	Data asynchronous transfert.	No
APEL [8]	Transfert links(Control flow or Data flow).	No
RHODES [21]	Function call.	No
App. for aquisition process architectures [17]	Objects that encapsulate communication mechanisms.	Yes
Connectors for bridging SP models [19]	Adaptation activities for particular products.	Yes
SPEM [1]	Links between "Work Product Ports".	No
App. based on evolution process components [3]	Communication unit.	Yes

Table II
CONNECTOR CONCEPT INTERPRETATION OF THE EXISTING APPROACHES.

approach defines its vision of the connector. We notice that the connector concept have more importance in the recent years; the recent approaches introduce the connector as first entity and give them more importance and functionalities.

Table -I- and table-II- summarize the architectural concepts identified from the discussed approaches.

B. Generic Metamodel For Software Process Architecture Description

To increase the flexibility of SPs, it is certain that defining explicit connectors has certainly a determining role on the SP modeling and execution quality. The possibility offered to specify, personalize, control, adapt, facilitate the interactions between SP activities is a major advantage. The transitions between activities can be controlled and the execution deviations limited.

We notice that process connectors can be identified to facilitate and adapt the products transfer [19] [17] Also, the controls flow aspect can be treated by explicit connectors, thus, it is possible to define connectors which evaluate the execution then decide changes to operate for better execution.

In our approach the process connector is treated as first class entity. We affect capital functionalities to them to allow specifying various data or control transfer kind. We define our process connector as an activity that "facilitate and control" transitions between the SP activities. The "process Connector" does not create new products, but "adapts and controls" existing products. The distinction between "creation" activities (which will constitute the process components) and "adaptation and control" Activities (which will constitute the process connectors) modifies the semantics of the identified concepts. Our interpretation of SP architectural concepts become as follow:

- **Process Component:** It corresponds to a "Work Unit" of the SP model. We describe a Process Component as treatments done on input products to "create" new products that are the output Products.
- **Process ports:** The Process Component interface is a set of "Process Ports", the required Process Port corresponds to the "input flow" needed to the execution of the SP component. The provided Process Ports is the "provided flow" or the Process Component output flow [1][3]. So, tow kinds of ports are defined:
  - **Data Flow Port :** That corresponds to the "Work Product Port" of the SPEM metamodel.
  - **Control Flow Port :** That is specific to the execution Control Flow of the SP models.
- **Process Connector:** A process connector is an Activity that "manages, adapts or controls" the execution of the SP. It is independent from the SP Method Content but depend on the execution or the structure of the SP. It facilitates the transmission of the Work Products or manages the Control Flow of the SPs. Some of our process connectors are inspired from APEL environment [8]. In fact APEL defines "adaptatives activities" that correspond to some of our Dataflow connectors (figure-3-).
- **Process Connector Role:** A connector interface is set of Process Connector Roles. It represents the flow (Data Flow or Control Flow) required or given by the Process Connector.
- **Binding:** Is a "link" between the internal Process Ports and the external Process Ports of the Process Component and Process Configuration. Definition of the

Software Process Concepts.	Adopted SP architectural concepts.
Activity that **creates** new products.	**Elementary Component**
Input or output flow of a creation Activity (DataFlow or ControlFlow).	**Process Port(given or required):** Can be a Control Flow Port or a Data flow Port.
Activity that **"adapts or controles"** the flow.	**Explicit Connectors:** Predefined connector's taxonomy.
Input or output flow of an adaptation Activity (DataFlow or ControlFlow).	**Process Connector Role:** Can be DataFlow Role Connector or ControlFlow Role Connector.
A set of "adaptation" and "creation" activities.	**Process Fragment:** An assembling of Process Components and Process Connectors.
Precedence links between a creation Activity and an adaptation Activity.	**Attachement:** A link between a Process Port and a Process Connector Role from the same kind (Control Flow or Data Flow).
Delegation links (Data Flow or Control Flow dalguation) between activities.	**Binding:** Links between Process Ports or between Process Connector Roles from the same kind (Control Flow or Data Flow)
Process structure.	**Process Configuration:** A set of process components and process connectors that respect assembling constraints.
Software life cycle, recurrent topological structure or execution strategy.	**Process style:** Introduced formally with type concepts, invariants and constraints.

Table III
ADOPTED SEMANTICS FOR SOFTWARE PROCESS ARCHITECUTRAL CONCEPTS.

binding concept allows describing the internal structure of the Process Configuration. As the same manner, the binding of the connector roles allows us defining the complex connectors and defining there internal structure by combining the predefined connector types.

- **Process configuration:** It describes the coherent assembly of the process components and process connectors by determining explicitly the connection and the constraints that must be respected. A configuration can follow a predefined style such as software life cycle or not.
- **Process Style:** It provides a partial and logic description of a SP assembly which has a predefined structure and that recur in a particular SP kind.

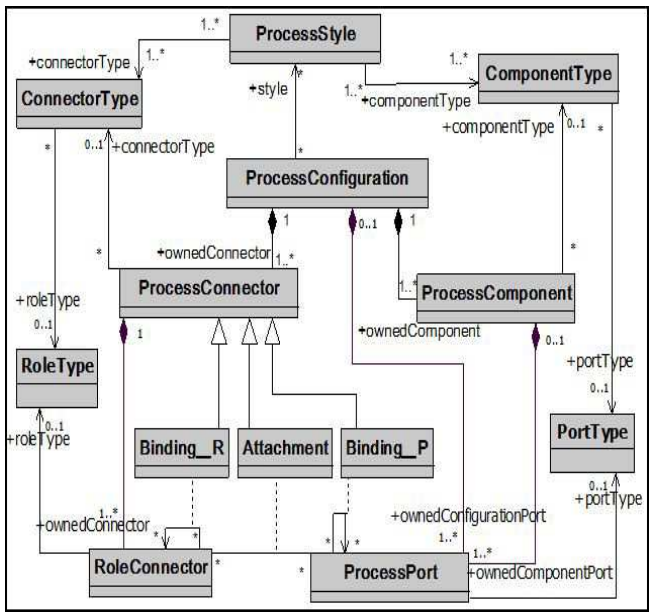

Figure 2. Generic metamodel for software process architectures.

We regroup the identified concepts in a generic metamodel based on UML (figure- 2-), this metamodel is independent from SPEM metamodel concepts and will be used to extend SPEM metamodel. We introduce the concepts "Component Type", "connector Type", "Port Type", "Role Type", these concepts are defined independently of the SPs concepts, and are introduced to describe SP styles formally.

C. Added Semantics to Process Connectors

By analyzing the SP models behavior we identify the recurrent activities of adaptation and Flow control. Thus, we identify taxonomy of explicit connectors for modeling SP architectures. These process connectors offer the possibility of managing the interactions independently of the SP kinds and orientation. These Process Connectors are very interesting for the agile methods that respect processes where flexibility and dynamicity are very required.

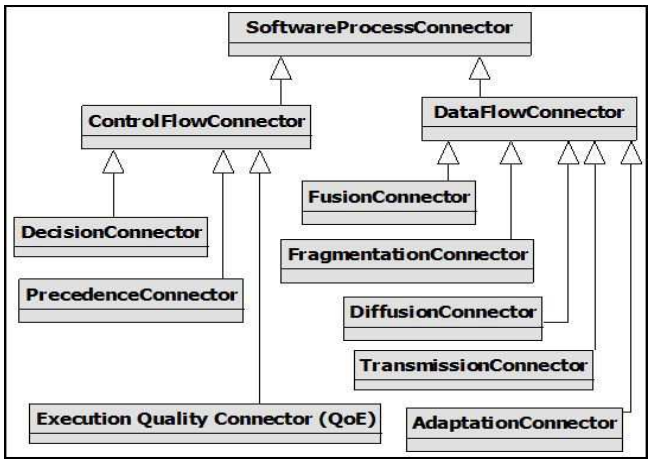

Figure 3. Software process connector's taxonomy.

According to Process Connector taxonomy, two kinds of process connectors are identified:

- **Data Flow connector:** That represents an activity that adapts the work product to be used by the connected Processe Components. This kind of activities are independent from the development or management activities, their preoccupation can be implemented and reused independently from the Process Components preoccupation. These connectors connect only Data Flow Ports.
- **Control Flow Connector:** They assure and control the execution quality of the SP model considering the stakeholders priorities. Defining these connectors give us the required SP flexibility. Introducing "Control Flow Connectors" that formalize the SP execution, allow not only personalizing the SP style execution, but also controlling them and handling the execution deviations by adapting the Process Connector parameters.

D. Added Semantics to Process Styles

Another important contribution is the formal introduction of SP styles. Defining architectural styles for SPs facilitates not only their modeling by exploiting the characteristics of the recurrent structures, but also, allows creating new SP models by combining different styles.

The identification of invariants, constraints of the recurrent structure of a given process allows extending this solution to other processes kinds that are not necessarily SPs.

Figure - 4- depicts the architectural view of the SP according to our adopted semantic. The Process Configuration is an assembly of Process Components and Process Connectors. The Control Flow is ensured by connectors "Control Flow" (Precedence Connector), on the other hand, the transfer of the work products is ensured by connectors

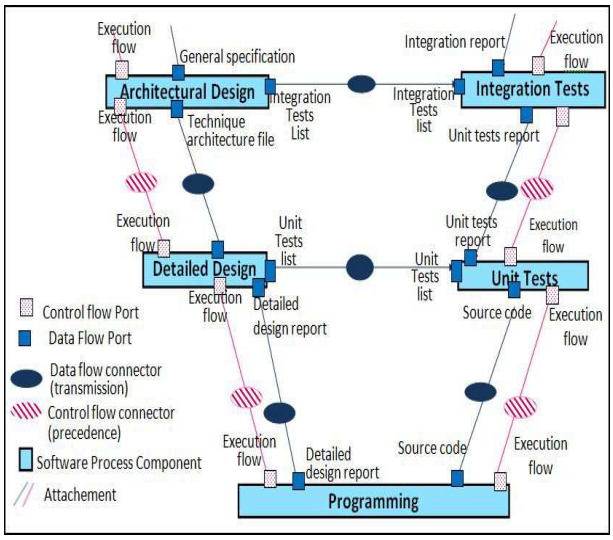

Figure 4. Process configuration respecting the ”V life cycle” topological style.

”Data Flow”(transmission connectors). These two kind of connectors have their own kind of Process Ports.

The interpretation of the software life cycle as architectural style is undeniable. The added semantics to the SP architectural concepts, allow not only defining ”topological” styles (V life cycles for example), but also, defining execution styles by adjusting the parameters of the ”Control Flow” connectors defined in our taxonomy.

Thus, the Process Configuration depicted in figure -4- respecting the topological style ”V life cycle” can be combined not only to other topological styles, but also, with other execution styles by introducing other types of ”Control Flow ” connectors. For example, by introducing Control Flow connectors ”Quality of execution (QoE)”, this configuration can be modeled to give the priority to time, the cost or the quality of the realization according to the stakeholders requirements.

V. Method Plugin Profile Extension

Having a complete semantic to describe a SP architecture, the extension of Method Plugin profile can be done, for this purpose, we introduce new stereotypes to describe the architectural elements of the SP architectures: Two abstract stereotypes are introduced (figure-5 -):

- **Process Architectural Element:** is an abstract ”Process Element” that describes the common characteristics of the process architectural elements that compose the structural view of the SP architecture.
- **Method Content Architectural Element :** is an abstract ”Method Content Element” that describes the common behavior of the process style architectural elements.

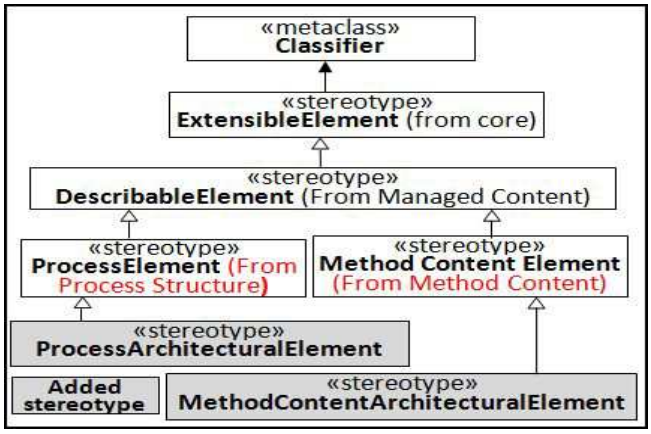

Figure 5. Method Plugin profile Extension with the abstract architectural concepts.

We distinguish these two stereotypes as the ”Process Style” is a ”Method Content Package” (Figure -6-) and its elements are only a ”Method Content Elements”, however, The ”Process Configuration” is a ”Process Package” (figure-7-) and its elements are only a ”Process Elements”.

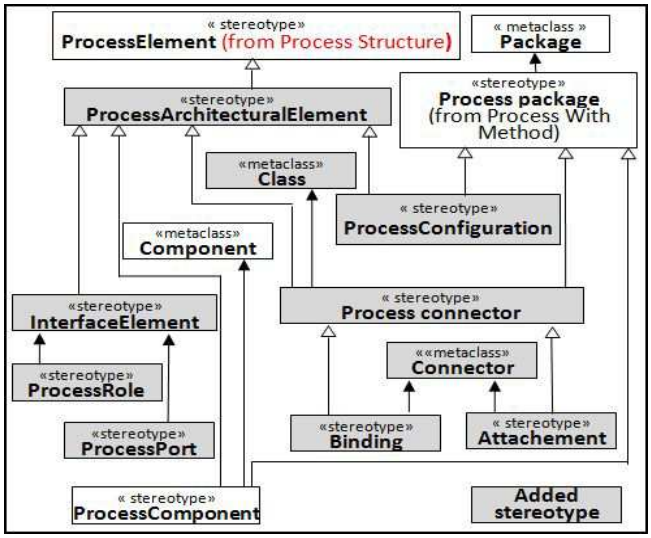

Figure 6. Method Plugin profile Extension with Process Architectural Elements.

The stereotypes specialized from ”Process Architectural Elements” describe the basic process configuration elements (Figure-6-). The ”Process Component”, ”Process Connector” and ”Process Configuration” describe treatements done to create or adapt process flow (DataFlow or ControlFlow). The ”Process Connector Roles” and the ”Process Ports” are interfaces of these treatements units.

The stereotypes specialized from ”Process Architectural Elements” describe the process style basic elements (Figure -7-). The ”Process style” stereotype is a ”Method Content Architectural Element” that describes the characteristics of

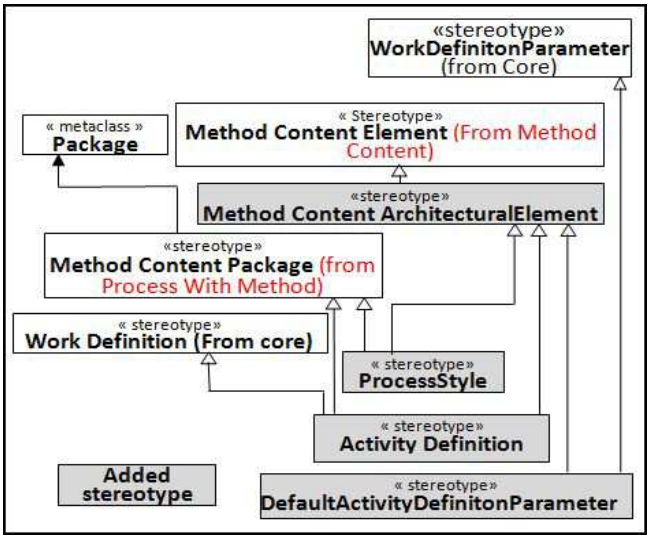

Figure 7. Method Plugin profile with Method Content architectural Elements.

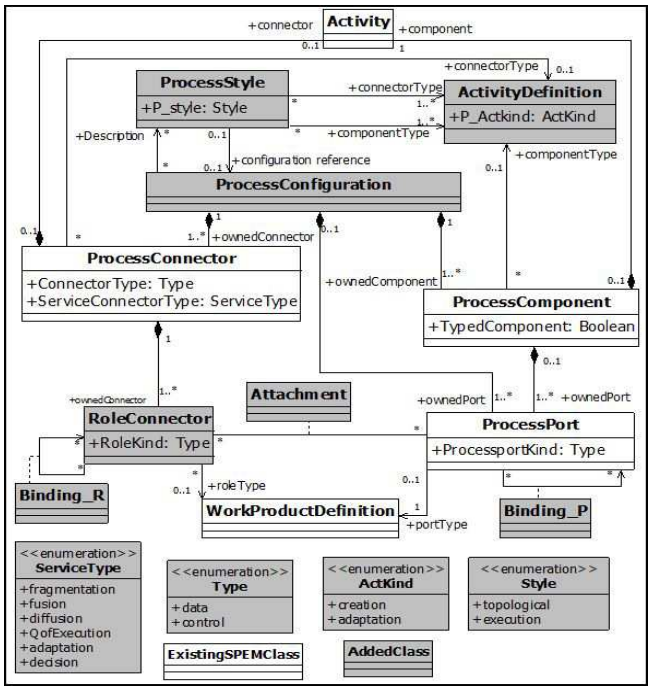

Figure 8. SPEM classes for describing Process Configurations.

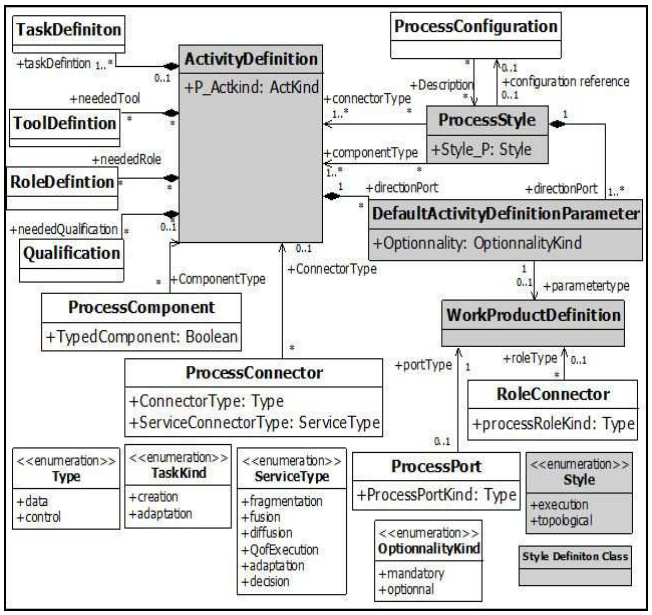

Figure 9. SPEM classes for describing processes styles.

the recurrent structures.

As ”Process Component” and the ”Process Connector” are activities, so the ”Process Activity Definition” is a ”Method Content Architectural Element” that describes the process component type and the process connector type. The ”Default Activity Definition Parameter” Is a ”Method Content Architectural Element” that describes the defaults parameter of the Process Activity Definition, it is used particularly to declare default directions of the Work Product Definition.

The figure -8- depicts the proposed SPEM metamodel to describe SP configurations. As software configuration, a SP configuration is composed from ”Process Components”, ”Process Connectors” and its interfaces. A ”Process Configuration” can respect ”Process Styles”. The ”Process Components” are connected by using explicit ”Process Connectors”. The link between ”Process Ports” of the ”Process Components” and ”Process Connector Role” of the ”Process Connector” is done via ”Attachments”.

We define a ”Process Style” according to its ”Activity Definition” that describes ”Process Connector Type” and ”Process Component Type”. At this level, only the ”Activity kind” property distinguish between the ”Process Component” and the ”Process Connector”. We Add ”Style kind” property to ”Process Style” class to differentiate between ”topological style” and ”execution style”.

The ”Activity” is constituted from a set of ”Break Down Element” as ”Task Uses”, ”Role Use”...etc. So, by transition, the ”Process Component” and ”Process Connector” (that are activities) are described with a set of ”Task Definition”, ”Role Defintion” ,”Tool Definition” and ”Qualification”. So, the ”Activity Definition” is composed from ”Task Definition”, ”Role Defintion” ,”Tool Definition” and ”Qualification” (figure-9-).

The ”Default Activity Definition Parameter” is used to describe the ”Work Product Definition” (that define the Process Port type and Process Pole Connector Type)Direction.

VI. Conclusion

This paper presents a complete semantic of the basic architectural concepts needed to describe SP architectures.

It treats also, the SPEM profile extension with the identified architectural concepts. This work is a necessary step to implement our proposed approach for reusing SPs based on software architectures. The proposed approach for reusing SPs is based on describing SP models as software architectures by exploiting SPEM metamodel conceptualization.

In order to extend the Method Plugin profile, we initially propose a generic metamodel for SPs architecture concepts. To identify the needed SP architectural concepts, we have identified the lacking architectural concept in SPEM metamodel. Then, we have studied the existing formal metamodels describing software architectures. As no complete formal metamodels for software process architectures is done, we strongly inspired from existing metamodels dedicated for software architectures [15] [14].

Through the analysis of SPEM packages and by considering the existing architectural concepts; we identify all SPEM concepts that can represent, specify, and specialize the identified architectural concepts or its properties. We proposed an extension of the Method Plugin profile. We detailed the added stereotypes and propose a SPEM extension to describe the process architectures and their styles.

Our most important contribution is the definition of explicit SP connectors; these connectors allow facilitating, adapting, controlling SP interactions.

The distinction between the "creation" activities and the "adaptation and control" activities confers the SP a greater flexibility and dynamicity by controlling and specifying interactions between activities which are much requested in the new development processes. It is considered as the basis of our software architecture definition; this distinction is motivated by the need to have an explicit execution model easily identifiable in the process model. This view allows controlling and SP execution and modifications. A process connectors' taxonomy is defined by studying the SPs behaviors; also, by composition we can create new connectors useful to specific situation that was not predict.

Other contribution is the definition of architectural styles specific to SPs, it contributes significantly to the SP modeling and the execution quality; thus, it is possible, not only to reuse the good practices and the particular strategies adopted by the processes developers (by formalizing this knowledge), but also, to combine, personalize, adapt these practices by ensuring a coherent results. Also, the "topological" and "execution" styles separation allow formalizing the execution view explicitly independently from the other views. This possibility is a significant advantage as it ensures the good execution of the SP model by allowing identifying errors easily and doing SP modifications rapidly, that ensure the success of the software development project. Defining formally invariants and constraints process styles is one of the perspectives of our work, for this purpose we are studying software life cycles characteristics to describe it formally.

By adopting the semantic suggested for the description of SPs architectures, it is possible to describe SP architectures in a rigorous way using existing ADLs. Thus, we could use ACME ADL to describe SPs Architectures without large difficulties. The suggested semantic formalize explicitly dependencies between Process Components, it handles the interconnection problems between Process Components by using explicit Process Connectors, which is more efficient compared with the semantic offered by SPEM metamodel. The integration of these concepts to SPEM metamodel is done by respecting the SPEM essence, the "Method Content" concepts are separated from "Process Structure" concepts. Also, it completes the SPEM purpose which is being a metamodel that describes the most kind of SPs including SPs architectures.

REFERENCES

[1] Object Management Group, *Software Systems Process Engineering Meta Model, v2.0*. http://www.omg.org/cgi-bin/docFormal/2008-04-01.

[2] N. Medvidovic, R.N. Taylor, *A Classification and Comparison Framework for Software Architecture Description Languages*. IEEE Trans. Software Eng. 26 (1): 70-93. 2000.

[3] F. Dai, T. Li, N. Zhao, Y. Yu, B. Huang , *Evolution Process Component Composition Based on Process Architecture*. ternational Symposium on Intelligent Information Technology Application Workshops, Vol.00, pp. 1097-1100, 2008.

[4] B. Combemale, X. Cregut, A. Caplain, B. Coulette Towards, *a Rigorous Process Modeling with SPEM*.ternational Conference Entreprise Information System ICEIS'06. 2006. PP. 530-533

[5] B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, R. Madachy, *Using the WinWin Spiral Model: A Case Study*. Computer, vol. 31, no. 7, pp. 33-44, July 1998.

[6] A. Alti, T. Khammaci and A. Smeda, *Integrating Software Architecture Concepts into the MDA Platform with UML Profile*.Journal of Computer Science 3 (10): 793-802.2007.

[7] N. Medvidovic, D. S. Rosenblum, D. F. Redmiles and J. E. Robbins, *Modeling Software Architectures in the Unified Modeling Language*.ACM Transaction on Software Engineering and Methodology, Vol.11, N1, page 2-57. January 2002.

[8] S. Dami, J. Estublier, M. Amiour, *APEL: a Graphical Yet Executable Formalism for Process Modeling*. journal of Automated Software Engineering, vol 5,pp 61-96. 1997.

[9] C. Hofmeister, R. L. Nord, D. Soni, *Describing Software Architecture with UML*. Proc. of 1st Working IFIP Conference on Software Architecture, pp 145-160,1999.

[10] M. Graiet, T. M. Bhiri, J.P. Giraudin, N. Belkhatir, *Architecture des systmes avec la norme UML2.0 et l'ADL Wright*. CAL2006, pp 83-100, 2006.

[11] J. Enrique, P. Martinez, *Heavyweight extensions to the UML metamodel to describe the C3 architectural style*. ACM SIGSOFT Software Engineering notes, vol 28, Issue 3, 2003.

[12] M. Goulao, F.B. Abreu, *Bridging the gap between Acme and UML 2.0 for CBD*. SAVCBS'03 Specification and Verification of Component-Based Systems, Workshop at ESEC/FSE, 2003.

[13] Object Management Group(OMG), *Unifed Modeling Language*. http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/.

[14] M. Mancona Kand ,A. Strohmeier, *Towards a UML Profile for Software Architecture Descriptions*. Proc. of the 3rd international conference on The unified modeling language, pp:513–527, 2000.

[15] A. Alti, A. Boukerram, A. Smeda, S. Maillard, M. Oussalah, *COSABuilder and COSAInstantiator: An Extensible Tool for Architectural Description*. J. of Software Engineering and Knowledge Engineering, pp 423-455, 2010.

[16] A. Hudaib, C. Montangero, *A UML Profile to Support the Formal Presentation of Software Architecture*. th Annual International Computer Software and Applications Conference, pp.217, 2002.

[17] J. Choi, W. Scacchi, *Modeling and Simulating Software Acquisition Process Architectures*. Journal of Systems and Software,vol 59, pp 343-354, 2000.

[18] N. Belkhatir, J. Estublier, *Supporting Reuse and Configuration for Large Scale Software Process Models*. th International Software Process Workshop, 1996.

[19] N. Medvidovic, P. Grünbacher, A. Egyed, B.W. Boehm, *Bridging models across the software lifecycle*. J. System Software, vol 8, pp 199-215, 2003.

[20] D. Avrilionis, N. Belkhatir, P.Y. Cunin,*A unified framework for software process enactment and improvement*. ICSP '96 Proceedings of the Fourth International Conference on the Software Process (ICSP '96), pp 102-108, 1996.

[21] B. Coulette, T. D. Thu, X. Crgut, , B. T. Dong Thi, RHODES: A Process Component Centered Software Engineering Environment. ICEIS'00, Second International Conference on Enterprise Information Systems, pp 253-260, 2000.