

# SPEM2XPDL: Towards SPEM Model Enactment

YUAN Feng<sup>1,3</sup>, LI Mingshu<sup>1,2</sup>, WAN Zhigang<sup>1,3</sup>

<sup>1</sup>Laboratory for Internet Software Technologies  
Institute of Software, The Chinese Academy of Sciences

<sup>2</sup>State key Laboratory of Computer Science  
Institute of Software, The Chinese Academy of Sciences

<sup>3</sup>Graduate University of the Chinese Academy of Sciences  
Beijing, China

**Abstract** - In recent years, enactment is being an important topic in the domain of software process. While SPEM, a standard metamodel put forward by OMG, gains increasing acceptance in industry, its model enactment is still at the early stage. In this paper, the SPEM2XPDL approach is proposed to support SPEM model enactment in the workflow paradigm. Based on a set of well-defined mapping rules, the transformation algorithm and engine are developed. This approach has been implemented in SoftPM project. After transformed to XPDL format, the SPEM models are successfully enacted by Shark, a famous XPDL-compliant engine. Finally, an example of the enactment of a review process is provided in this paper.

**Keywords:** SPEM, XPDL, Software Process, Workflow, Model Transformation.

## 1 Introduction

SPEM (Software Process Engineering Metamodel)[1] is the software process modeling standard put forward by OMG (Object Management Group) in 2002. It is a methodology-independent language based on UML. OMG's objective is making it the unified software process modeling standard in industry. Nowadays there are many process models defined in SPEM such as IBM's RUP, DMR's MacroScope and Unisys's QuadCycle. However, because the initial intention is to describe the software development process, there is the lack of process enactment supporting in SPEM [2].

Workflow is the automation of a business process during which documents, information, and tasks are passed from one participant to another for actions according to a set of procedural rules [3]. To support the definition, creation, management and execution of workflows, WFMSs (Workflow Management System) are developed. In order to standardize the interoperability of different capabilities of WFMSs, WfMC (Workflow Management Coalition) defines five interfaces for WFMSs. Among them, XPDL is the standard process definition interface that separates the workflow definition from execution. That is, once a model is XPDL conformed, it is able to be enacted by any XPDL-compliant engine. The applications on workflow execution are rather mature; WFMSs such as IBM's WebSphere MQ Workflow and open-source tool Shark [4] have already been widely used.

SPEM is expressive and strong in the software process modeling; while XPDL is strong in supporting the powerful WFMS execution mechanism. In this paper, we propose the SPEM2XPDL approach which gets the bilateral merits through the union of them. In our approach, at first, software processes are modeled in SPEM

metamodel; then the models are transformed to XPDL form by specific transformation engine; finally the resulted XPDL models are enacted on XPDL engines such as Shark.

This paper is organized as follows. Section 2 introduces the related work; In Section 3, a SPEM-modeled software process SoftPM\_RV is provided to be the example throughout this paper; In Section 4, the well-defined mapping rules of SPEM2XPDL are introduced in detail; The corresponding transformation algorithm is provided in Section 5; Section 6 introduces the application in SoftPM project, an enactment scenario of SoftPM\_RV is illustrated. Finally, the conclusion and the expectation of future work are given in Section 7.

## 2 Related Work

Since 1990s, the idea of using workflow paradigm as a basis to support the enactment of software process is proposed. From the investigations of [5][6], It is found that, although software processes have their own characteristics compared to general processes, for example, they are dynamic, hard to be pre-defined, need more elaborate control flows, etc., they both obey the same basic paradigm. In the workflow view, software process is a special kind of business process in which documents, information, and tasks are passed from one participant to another according to a set of development methodology related rules [7]. Thus the management and enactment of software process and workflow can be supported by the same mechanism [7][8].

Based on this viewpoint, there are many researches and practices in these years; these works provide the feasibility proof in turn. In [7], a meta-schema is provided as the software process metamodel while using RDBMS as the PSEE, the models are then transformed to the

XPDL ones to be enacted. In [9], the WFMS KAOS is used as the core of a PSEE. In [10], a software process enactment environment, PIEnvironment, is developed based on the commercial workflow system WebDeploy. In [11] and [12], UML modeled software processes are transformed to BPEL (Business Process Execution Language) /XPDL models aiming for the enactment.

In the above-mentioned works: 1) As Barnes et al mentioned, the mapping rules, cover comprehensive considerations, are still to be defined [7]. 2) Different software process modeling metamodels, whether implicitly or explicitly defined, are adopted and then different mappings to workflow are applied. These bring the negative impacts to the reusability and interoperability of their research results. In our approach, however, the metamodels SPEM and XPDL are both defined by the standardization organizations and widely accepted. Thus the reusability and interoperability are guaranteed.

### 3 A SPEM Model: SoftPM\_RV

SoftPM is a powerful toolset for software development, with a process management tool included [13]. In SoftPM there are many software processes defined in SPEM. To explain our approach, SoftPM\_RV, a review process in SoftPM, is chosen as the example throughout this paper.

Fig 1 illustrates part of SoftPM\_RV modeled using Enterprise Architecture. Modeled in SPEM, it is clear and understandable. In SoftPM\_RV, there are four process performers defined: "Project Manager", "Review Leader", "Review Member" and "Product Manager". Six activities are connected orderly by precedes relationships or intermediate workproducts. In this example, the "Prereview" activity is specified by another activity diagram which contains four sub-activities. Further on, the sub-activity "Independent Review" is described by four steps.

Enactment of the software process is the next expected step after modeling. The following basic scenario is our intention to be supported. Once a SPEM model is created as above, the control flow between activities could be managed efficiently, corresponding activities will be assigned to respective process roles according to the predefined sequence or/and specified constraint conditions. The control flow, task assignment, even the work product management should be automated as much as possible.

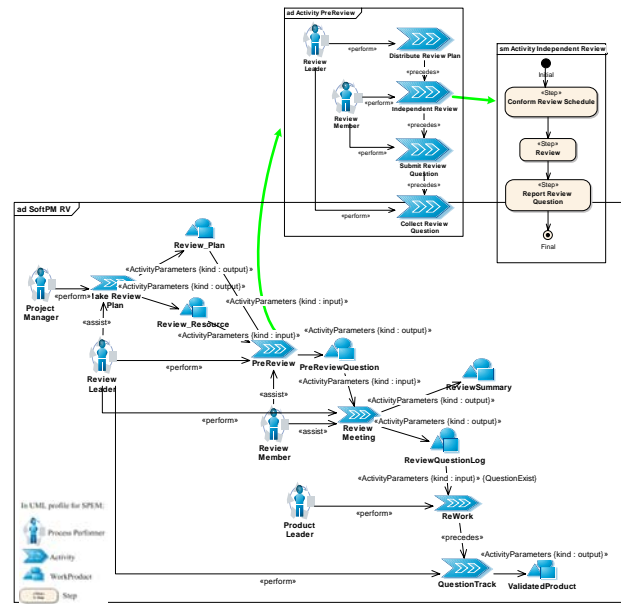


Fig1. The SoftPM\_RV Process Model(Partial)

However, as SPEM's initial intention is for description, there is not such supporting system developed yet. Developing such a system from scratch is both unpractical (immense effort and cost) and unnecessary. In our practice, we choose Shark, a XPDL engine, as the infrastructure to accomplish the objective, thus both the cost and time are highly reduced.

## 4 The SPEM2XPDL Approach

There are three main parts included in the SPEM2XPDL approach: 1) the mapping rules between SPEM and XPDL metamodels; 2) the corresponding transformation algorithm; 3) the transformation engine reflecting the algorithm. In this section, the mapping rules are introduced. They are grouped into two kinds: rules between entities and rules between relationships.

### 4.1 Rules between Entities

Table 1 lists the major entity-mappings. When an object in SPEM model is transformed, its attributes should be transformed in the mean time. Due to the lack of space, the complete mapping rules including metaclass attribute mappings are not provided in this paper.

Table 1. Major entity-mappings in SPEM2XPDL

No.	SPEM Element	XPDL Element	Description
R1	Package	Package	The extended attribute of <i>Package</i> in XPDL model will be set to distinguish different source SPEM elements. For example: <ExtendedAttribute Name="type" Value="Phase"/>
R2	Discipline	Package	
R3	Phase	Package	
R4	Lifecycle	Package	

R5	Process	WorkflowProcess	Transformed to the same name <i>Process</i> in XPDL model.
R6	Iteration	Activity	<i>Iteration</i> is a composite <i>WorkDefinition</i> with a minor milestone. The correspondence in XPDL model is an <i>Activity</i> whose implementation type is “subflow”. This <i>Activity</i> points to a sub-process which includes the iteration contents.
R7	Activity	Activity	If the <i>Activity</i> is further described using <i>Steps</i> , then the correspondence is a <i>Block Activity</i> points to an <i>Activity Set</i> , else is an atomic activity.
R8	Step	Activity	As stated above.
R9	WorkProduct	ExtendedAttribute	An activity-related input/output <i>WorkProducts</i> are transformed to the extended attributes of corresponding <i>Activity</i> .
R10	WorkProductKind	ExtendedAttribute	Transformed to the extended attribute of the “ <i>WorkProduct</i> ” extended attribute.
R11	ProcessPerformer	Participant	The <i>ParticipantType</i> of Participant is set to “Role”.
R12	ProcessRole	Participant & ExtendedAttribute	Transformed to corresponding Participant; Transformed to the extended attribute “assistant” of corresponding <i>Activity</i> .
R13	Guidance	ExtendedAttribute	Transformed to the extended attribute of <i>Activity</i> , for example: <ExtendedAttribute Name="Guidance" Value="X"/>
R14	Goal	ExtendedAttribute	Similar to <i>Guidance</i> .
R15	Precondition	Transition Condition	Set the <i>Precondition</i> to the condition of input <i>Transition</i> .

On the one hand, because software process is being treated as a specialization of the more general workflow, as we can see in Table 1, there are cases of several SPEM elements mapping to one, more general, workflow elements. On the other hand, while XPDL aims at the execution of workflow, as to many attributes of *Activity* in XPDL model, the corresponding source values in SPEM model cannot be found. In our approach, these attributes are given default values during transformation; for example, the activity attribute *Start Mode* is set to “Manual”, and the default *Transition Restrictions* is set to “AND Join” and “AND Split”.

During metamodel-level transformation, the metamodel inconsistency is one of the key problems. In this case, the major inconsistency is caused by *WorkProduct*. In SPEM model, *WorkProduct*, *Role* and *Activity* are the three core concepts. But in XPDL model, there is not the direct correspondence of *WorkProduct*. As listed in Table 1, in our approach, *WorkProducts* are specified as extended attributes of corresponding *Activities* in XPDL model. The transformation from first-class metaclass to the attribute attached to *Activity* brings the loss of some semantics. These semantics can be classified as:

1) The consistency of *WorkProduct* attributes.

Although all the attributes can be represented as the extended attributes in XPDL model with the same names,

their consistency cannot be maintained automatically. For example, the “state” attribute of a *WorkProduct* will appear separately in the *ExtendedAttribute InputWorkProduct* and *OutputWorkProduct* of two *Activities* at the same time, when one of them is changed, the another one can only be changed synchronously manually.

2) The associations between *WorkProduct* and other metaclasses. For example, the association *WorkProduct::responsibleRole* between *WorkProduct* and *ProcessRole*.

3) The attributes of other metaclasses which are related to *WorkProduct*. For example, the attribute *hasWorkPerArtifact* of *WorkDefinition*.

Although these semantics are lost, there are still enough semantics that are preserved which can support the enactment on XPDL-compliant WFMS. In other words, the transformation does not aim to give a substitute of SPEM model, but to provide an approach supporting its enactment.

## 4.2 Rules between Relationships

In XPDL, there is only one relationship between two activities, the transition relationship. But in SPEM, it is much more complex. We group them as four kinds:

1) Layer Relationship (LR)

A LR maybe: a) the *ownedElement* association between *Package* (and subclasses of *Package* such as *Process* and *Discipline*) and *ModelElement*; b) the association between *Activity* and *Step*. The correspondence of LR in XPD model is *Package* or *ActivitySet: Activity*.

*Definition 1.*

$$LR = \{r \mid (r \in R) \wedge (r.oclIsKindOf(Association)) \wedge P(r, CLR)\}$$

R stands for the universal set of the relationships in SPEM, *oclIsKindOf* is the keyword in OCL (Object Constraint Language) [14], *r.oclIsKindOf(Association)* determines whether *r* is an *Association*. Predication  $P(r, CLR)$  determines whether *r* conforms to the constraint condition CLR. CLR is specified using OCL as follows:

```
[CLR] context r inv:
self.associationEnd->size=2
and
((associationEnd-
>forAll(a1,a2|a1.oclIsKindOf(Package)
implies a2.oclIsKindOf(ModelElement) ))
or
((associationEnd->forAll(a1,a2|a1.oclIsKindOf(Activity)
implies a2.oclIsKindOf(Step) )))
2) Role-Activity Relationship (RAR)
```

A RAR maybe: a) the *Perform* relationship between *WorkDefinition* and *ProcessPerformer*; b) the *Assistant* relationship between *Activity* and *ProcessRole*. In XPD model, the former determines the activity participants; the latter is corresponding to the extended attributes of activities.

*Definition 2.*  $RAR = \{WorkDefinition::Performer, Activity::Assistant\}$ .

3) Transition Relationship (TP)

TP is the Relationship between the same level entities. It maybe *Precedes*, or the relationship compositions “Activity-WorkProduct-Activity (AWAR)” in SPEM model.

*Definition 3.*  $AWAR = \{(a1, w, a2) \mid P(a1, w, a2), CAWAR\}$ .

```
[CAWAR] context (a1, w, a2) inv:
(a1.oclIsKindOf(Activity) )
and (a2.oclIsKindOf(Activity))
and (w.oclIsKindOf(WorkProduct))
and ((a1->exitsts (OutputWorkProduct=w) )
and (a2->exitsts (InputWorkProduct=w) ) )
```

*Definition 4.*  $TR = \{Precedes, AWAR\}$ .

CAWAR requires that the *WorkProduct* *w* is the input of *Activity* *a1* and the output of *Activity* *a2* at the same time. The corresponding in XPD model is the transition between two activities.

4) Other Relationship (OTR)

Except for what defined above, all the other relationships are defined as OTR.

*Definition 5.*  $OTR = \{impact, trace, referto, categorizes, govern\}$

Since in XPD model, there is only transition relationship between activities. Some relationships,

especially the OTR ones such as *trace*, *impact* relationships between work products, will be lost during the transformation. However, as mentioned above, the transformed semantic is enough to support its enactment by WFMS.

## 5 Transformation Algorithm

The main idea of our algorithm is: firstly, group the related elements in the same namespace according to the LR relationships; secondly, transform the entities in each namespace according to specified rules; some steps, however, requires manual intervention.

As illustrated in the algorithm, firstly, an empty XPD model is created and then initialized with Process Header and Redefined Header; in the third step, a mapping list *m\_list* is created, in each transformation step followed, this list is updated; fourthly, all the roles in SPEM model are transformed into corresponding participants; fifthly, empty WorkflowProcesses is created, the top-level namespace elements will be filled here later; in the sixth step, the elements of SPEM model in same namespaces are grouped respectively; they are then transformed according to corresponding rules in the seventh step; while some semantics such as constraint conditions in SPEM model are not transformed automatically, they are added manually in the eighth step; finally there is a process validity checking.

*SPEM2XPD Algorithm:*

Input: SPEM model    Output: XPD model

- (1) Create an empty XPD Model;
- (2) Create Package Header and Redefinable Header;
- (3) Create *m\_list*; // the mapping list
- (4) Transform all roles by [R11][R12]: {process performers, process roles} -> {Participant}
- (5) Create empty WorkflowProcesses;
- (6) For each association *r*
  - (6.1) If *r* LR // matching {Package, Discipline, Phase, Lifecycle, Process, Step}
    - (6.1.1) Store all SPEM elements according to namespaces;
    - (6.1.2) For each namespace,
      - (6.1.2.1) If not the top-level namespace, create corresponding ActivitySet in ActivitySets of the resulted XPD model;
      - (6.1.2.2) Goto (7);
- (7) For the elements in one namespace
  - (7.1) Transform all activities by [R5][R6][R7][R8]: {Iteration, Activity, Step} -> {Activity};
  - (7.2) For each association *r*
    - (7.2.1) if *r* RAR
      - (7.2.1.1) if is (WorkDefinition::Performer), set the “Performer” value of corresponding activity;
      - (7.2.1.2) if is (Activity:Assitant.), set extended attribute of corresponding activity by [R12].
    - (7.2.2) if *r* TR // matching {precedes, ActivityParameters {kind: input/output}}

- (7.2.2.1) Create transitions: {TR}-> {transition};
- (7.2.3) if r OTR
- (7.2.3.1) if r is {govern}, create corresponding extended attributes to the activity.
- (7.3) For other element e
  - (7.3.1) if e {Guidance, Goal, Precondition}, set extended attribute to corresponding activity by [R13][R14][R15];
  - (7.4) Add “Start”/“End” extender attributes and route activities;
- (8) Specify the constraint conditions manually;
- (9) Process validity checking;
- (10) Algorithm ending;

## 6 Enacting SoftPM\_RV based on Shark

According to the algorithm, we develop an SPEM2XPDL transformation engine. Its input is the XML file of SPEM model, which can be exported by SPEM modeling tools such as Enterprise Architecture. The output is the XML file of the resulted XPDL model, which can be enacted by XPDL compliant tools such as Shark.

In SoftPM project, we transform SPEM modeled software processes to corresponding XPDL format, and then successfully enact them to guide the software development. In this way Shark serves as a good infrastructure framework for the management of software development processes, especially the job scheduling and personnel allocation.

Take SoftPM\_RV as the example, Fig 2 is the process view in Shark after the resulted XPDL model is loaded. The nested activity diagram in SPEM is transformed to corresponding sub-flow; the steps are transformed to corresponding activities in specified ActivitySet. Due to the Shark restriction, route activities such as “start0”, “start1” and “end0”, “end1” are added.

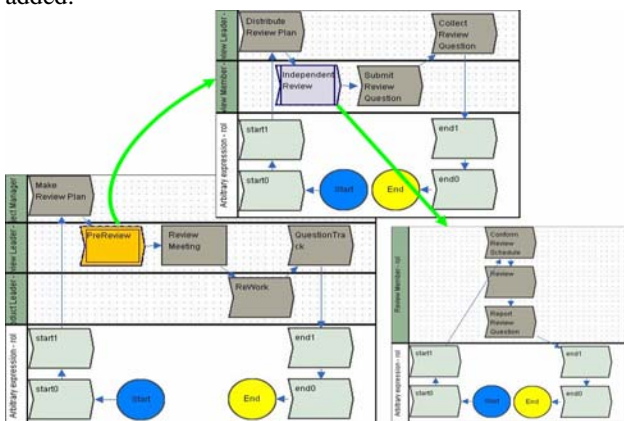


Fig 1. The SoftPM\_RV enacted by Shark

Configuration step is needed before enactment: 1) user management; the mappings between users and participant roles are defined. For example, user “Feng” is set as the “Project Manager”; 2) application declaration; through this step, supporting tools are well integrated with

WFMSs. In SoftPM project, we developed tools such as product management and data collection applications. As an example, after the data collection application is integrated into Shark, it records the efforts of every task automatically.

After configuration, a typical enactment scenario is: Firstly, when each role login the system, he will get a specific task list. In this example, when project manager “Feng” login the system, he will get the first task “Make Review Plan”, as illustrated in Fig 3.



Fig. 3 The task list of the user “Feng”

When he finishes this job, the data collection application is called automatically to record his effort on this task. In the mean time, the subsequent activity “PreReview” is assigned to the review leader. When he login and start this task, the sub-flow “PreReview” is started up and corresponding tasks “Distribute Review Plan” will be assigned. Again, the effort will be automatically recorded when these tasks are finished.

As illustrated in this example, the control flow in software process is successfully managed through the workflow paradigm. Meanwhile, software process specific supporting tools are integrated through the standard WFMS interfaces. In addition to the data collection application, a set of other applications focus on different aspects of software process management are also developed and efficiently implemented in SoftPM project, for instance the resource organization and workproduct management applications.

## 7 Conclusions

In this paper, we aim to support the enactment of software process in the workflow style. Our approach is built based on SPEM and XPDL standards, they are both defined by the standardization organizations and widely accepted, this guarantees the reusability and interoperability of our approach. Specifically, the mapping rules from SPEM to XPDL are well defined; corresponding transformation algorithm and engine are developed and successfully implemented in SoftPM project.

Through the combination of SPEM and XPDL standards, the major benefits of SPEM2XPDL approach are: 1) the reusability and interoperability gained from SPEM being the industry standard; due to the reusability feature. the time and cost of developing the support system may be highly reduced; 2) the openness and flexibility brought by the general purpose paradigm of

workflow; 3) the methodology-independence as SPEM is adopted, different software development processes such as RUP, XP and QuadCycle, once it can be modeled in SPEM, are supported by our approach.

Besides SoftPM project, this approach has also been used to enact part of RUP, a famous software development process model. In the next phase, we are intended to apply it to more process models and further consummate the transformation rules according to different situations. Besides, as there is no explicit constraint specification mechanism in the present version of SPEM, the constraint conditions such as the choice/branch structure definition in XPDL model is supplied manually. As OCL is to be adopted in the coming version of SPEM [2], another future work will be focused on the parse and transformation of OCL expressions, it will further raise the automatic degree of our approach.

## 8 References

- [1] OMG, Software Process Engineering Metamodel Specification. Version 1.0 (formal/2002-11-14), <http://www.omg.org>, 2002.
- [2] OMG, Software Process Engineering Metamodel (SPEM) 2.0 RFP (ad/2004-11-04), <http://www.omg.org>, 2004.
- [3] WfMC, XML Process Definition Language Specification (Draft 1.0), Document Number WfMC-TC-1025, <http://www.wfmc.org>, 2002.
- [4] Shark, <http://shark.enhydra.org>, 2005.
- [5] Chan, D.K.C and Leung, K.R.P.H, "A Workflow Vista of the Software Process," Proceedings of the International Workshop on Database and Expert Systems Applications. Toulouse, France, pp. 62-67, September 1997.
- [6] Chroust, G, "Interpretable Process Model for Software Development and Workflow," Proceedings of the European Workshop on Software Process Technology. Noordwijkerhout, The Netherlands, pp. 144-153, April 1995.
- [7] Barnes,A and Gray, J, "COTS Workflow and Software Process Management: An Exploration of Software Engineering Tool Development," Australian Software Engineering Conference, Queensland, Australia, pp. 221-234, April 2000.
- [8] Chan, D.K.C and Leung, K.R.P.H, "Software Development as a Workflow Process," Proceedings of ICSC, Hong Kong, China, pp. 282-291, December 1997.
- [9] Penadés, M.C and Canós, J.H and Carsí, J.A, "Workflow Support to the Software Process," MENHIR'99, 4th MENHIR Workshop, Sedano, <http://giro.infor.uva.es/workshop4/Presentaciones/Sesion2/Workflow.ppt>, 1999.
- [10] Nitto, E.D and Lavazza, L and Schiavoni, M and Tracanella, E, "Deriving executable process descriptions from UML," Proceedings of the 24th International Conference on Software Engineering, Buenos Aires, Argentina, pp. 155-165, May 2002.
- [11] Keith, M, "From UML to BPEL, Model Driven Architecture in a Web services world." <http://www-128.ibm.com/developerworks/cn/webservices/ws-uml2bpel/index.html>, 2003.
- [12] Jiang, P and Mair, Q and Newman, J, "Using UML to Design Distributed Collaborative Workflows: from UML to XPDL," IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Linz, Austria, pp. 71-76, June 2003.
- [13] Wang, Q and Li, M.S, "Software Process Management Practices in China," Proceeding of the Software Process Workshop (SPW2005), Beijing, China, pp. 317-331, May 2005.
- [14] OMG, OCL 2.0 Final Adopted Specification (ptc/2003-10-14), <http://www.omg.org>, 2003.