An object-oriented description method of EPMM process

Zuo Jiang$^{1,2,3}$ and Fan Yang$^{1,3}$

$^1$ School of Mathematics and Computer Science, Yunnan Minzu University, Kunming, Yunnan 650504, China
$^2$ School of Software, Yunnan University, Kunming, Yunnan 650504, China
$^3$ Key Laboratory of IOT Application Technology of Universities in Yunnan Province, China

Abstract. In order to use the object-oriented mature tools and language in software process model, make the software process model more accord with the industrial standard, it’s necessary to study the object-oriented modelling of software process. Based on the formal process definition in EPMM, considering the characteristics that Petri net is mainly formal modelling tool and combining the Petri net modelling with the object-oriented modelling idea, this paper provides this implementation method to convert EPMM based on Petri net into object models based on object-oriented description.

1 Introduction
In recent years, more and more attention is focused on the research of software process based on the idea of object-oriented. Because the object-oriented method has the mature industry standard and obtains a lot of tools to support, the software process combining with the idea of object-oriented is more standardized and more easy software implementation. Hsueh et al.[1] proposed an approach based on UML (Unified Modelling Language) to define, validate and confirm the software process. Hsueh et al. also use OCL (Object Constraint Language) to define the limits and rules that the models must meet. Lee et al.[2] proposed two methods about software process modelling, the first one based on UML meta model was to define the tasks allocation strategy in software process, and the second was to check inconsistency with the process description based on UML. OMG (Object Management Organization) has proposed SPEM (Software Process Engineering Meta-Model), and the SPEM2.0 was released in 2008 later[3]. UML is the SPEM main definition language. SPEM is suitable for various types of software process life cycle model, and it is a general framework, so SPEM has become the industry standard for software process modelling. However for the sake of readability and simplicity, SPEM does not use a formal language to model constraints.

In order to support both software evolution modelling and process modelling, Li[4] first proposed EPMM (Evolution Process Meta-Model) which uses completely formal language (Petri net and Hoare logic), by which process elements such as tasks, activities, and processes are defined. EPMM can be convenient to solve these problems which are appearing in software evolution process, such as uncertainty, hierarchy and concurrency etc.[4]. However, based on EPMM software evolution modelling and process modelling currently lack the support of effective popular tools, and EPMM modelling application has a lot of limitations. In order to extend the EPMM modelling application, it is a feasible method to convert EPMM into object-oriented description in an popular object-oriented Language.
From the object-oriented perspective, according to concepts in SPEM defined by OMG and using executable UML and OCL, this paper provides a method to convert EPMM into object-oriented description, and attempts to provide wider applications of the software process modelling and a new way of the future research.

2 EPMM elements object-oriented description

The corresponding relation between EPMM and SPEM is described as follows: (1) The concepts of tasks, activities and software processes in EPMM are corresponding to the Task class, Activity class and Process class in SPEM; (2) EPMM uses Petri net to express the sequence of activities and the relationship between the activities. However, SPEM uses the association class WorkSequence to describe the order relations between the active entities. Association class WorkSequence has 4 sequence relations: startToStart relation, startToFInish relation, finishToStart relation, finishToFInish relation.

2.1 EPMM object-oriented description

The Task class, Activity class and Process class in SPEM indirectly inherit from the extension class defined by metaclass Class, Classifier and Action[3]. The detailed definition of meta classes, extension classes, association classes and inheritance relationship between classes in SPEM can be found in [3]. Based on the characteristics of software evolution process and the simplicity of the meta model, this paper only considers the object-oriented description of Task class, Activity class and Process class and their relationship. figure 1 is the main structure of EPMM object-oriented description.

![Figure 1. The main structure of EPMM object-oriented description.](image)

The task element in [4] is defined as four-tuple that task \( t = \langle \{ Q_1 \} , \{ Q_2 \} , M_t , M_o \rangle \). In object-oriented description, Task element in EPMM corresponds to the Task class in object-oriented description, component pre assertion \( \{ Q_1 \} \) and component post assertion \( \{ Q_2 \} \) in task \( t \) four-tuple correspond respectively to the property preCondition and property postCondition in class Task, and component \( M_t \) that expresses sending messages and component \( M_o \) that expresses receiving messages correspond respectively to the property receivedMessages and property sentMessages in class Task.

The activity element in [4] is defined as four-tuple that activity \( a = \langle I , O , L , B \rangle \). Activity element in EPMM corresponds to the Activity class in EPMM object-oriented description. In activity \( a \) four-tuple, component \( I \) that expresses input data structure, component \( O \) that expresses output data structure and component \( L \) that expresses local data structure correspond respectively to the property inputData, property outputData and property localData in class Activity. In addition, in order to realize 4 sequence relations (startToStart, startToFInish, finishToStart, finishToFInish) in SPEM, the startTime and finishTime properties are added to describe the start time and finish time of the activity in object-oriented description.
The process element in [4] is defined as four-tuple that process $p = (C, A; F, M). Process$ element in EPMM corresponds to the $Process$ class in EPMM object-oriented description. In process $p$ four-tuple, component $C$ that expresses condition sets, component $M$ that expresses case sets correspond respectively to the $conditionSet$ and $caseSet$ in class $Process$; component $A$ that expresses activity correspond to the $Activity$ class in EPMM object-oriented description, in addition, component $F$ that expresses flow relationship between activities is realized by instantiating the Association class $WorkSequence$ in EPMM object-oriented description.

Based above analysis, we can see that from the static structure, the main structure of EPMM object-oriented description can descrit correspondingly EPMM structure defined in[4].But EPMM not only has the static structure but also has the dynamic behaviour. In addition, UML used in EPMM object-oriented description is not a complete formal language, so further extending and formalizing EPMM object-oriented description is necessary. In order to achieve this goal, this paper also complements the EPMM object-oriented description from the aspects of abstract syntax and object constraint semantics.

2.2 Abstract syntax of EPMM object-oriented description
Any modelling language has a set of modelling language specified in the text form and has text models in text format, so the text format is generally referred to as the model abstract syntax[5]. In order to enrich the semantic of EPMM object-oriented description, EBNF is used in this paper.

$Process::=\langle processID, conditionSet, caseSet, \{activities\} \rangle$
$Activity::=\langle activityId, inputData, outputData, localData, startTime, finishTime, tasks, \{linkToSuccessor\}, \{linkToPredecessor\} \rangle$
$Task::=\langle taskId, preCondition, postCondition, receivedMessages, sentMessages, execute \rangle$
$WorkSequence::=\langle kind, Successor|Predecessor \rangle$
$WorkSequenceKind::=\langle finishToStart|finishToFinish|startToStart|startToFinish \rangle$

**processID** ::= URI
**conditionSet** ::= quotedString
**caseSet** ::= quotedString
**activities** ::= activityId
**activityId** ::= URI
**inputData** ::= quotedString
**outputData** ::= quotedString
**localData** ::= quotedString
**startTime** ::= quotedString
**finishTime** ::= quotedString
**tasks** ::= taskId
**linkToSuccessor** ::= kind
**linkToPredecessor** ::= kind
**taskId** ::= URI
**preCondition** ::= quotedString
**postCondition** ::= quotedString
**receivedMessages** ::= quotedString
**sentMessages** ::= quotedString
**execute** ::= quotedString
**kind** ::= WorkSequenceKind

URI ::= a string of characters used to identify a name or a resource on the internet. One can classify URIs as locators (URLs), oras names (URNs), or as both. The URN defines an item's identity, while the URL provides a method for finding it. [PAN10]

quotedString ::= a nonempty finite sequence of characters in which “ (U+22) and \ (U+5C) occur only in pair of the form “ (U+5C, U+22) and \\ (U+5C, U+5C), enclosed in a pair of ” (U+22) characters, i.e. a Unicode string in normal form C. [PAN10]
The above abstract syntax is only one of definition methods. In order to ensure the simplicity of syntax, some corresponding properties of element components in EPMM (such as \{Q_1\} and \{Q_2\} in task $t$, I and O in activity $a$) just make simple definitions. In above abstract syntax, properties preCondition and postCondition in class Task, properties inputData, outputData in and localData in class activity are all simply defined as strings.

3 Abstract syntax of EPMM object-oriented description

Following EPMM, the software process models in literature[4] are to model by Petri net tools, and process models object-oriented description in this paper are to model by UML class diagrams. There are four important basic process blocks defined in literature[4], which are all described by Petri nets. Process basic blocks are as follows: sequence block, concurrency block, selection block and iteration block. These basic blocks are the basis of constructing process models by stepwise refinement method. In this paper, we take the four basic blocks as an example to illustrate how to implement object-oriented description about process models.

3.1 Sequence block object-oriented description
In the literature[4], the process sequence block consists of two activities $e_1, e_2$ and the process model described by Petri nets is shown in figure 2(a). Accordingly object-oriented description described by UML class diagrams is shown in figure 2(b), among which, there are two activity objects $e_1, e_2$ and one relation object finishToStart.

![Figure 2. The object-oriented description of process sequence block](image)

The object-oriented description of process sequence block is based on UML. However UML is not a complete formal language, so it is necessary to extend the semantics and further formalization. This paper uses the OCL formal expressions(such as invariant, pre/post operation conditions, rules, etc.) to supplement semantic constraints.

Due to the association relationship between activity $e_1$ and $e_2$ is finishToStart relationship in the object-oriented description of process sequence block, in terms of the active object $e_1, e_2$ association relationship, the following constraints written in OCL are established.

\[ e_1.linkToSuccessor.kind==WorkSequenceKind.\text{finishToStart}\implies e_1.\text{finishTime} \text{isBefore}(e_1.\text{linkToSuccessor.\text{startTime}}) \]
3.2 Concurrency block object-oriented description
In the literature[4], the process concurrency block consists of four activities e0,ei,ej,en and the process model described by Petri nets is shown in figure 3(a). Accordingly object-oriented description described by UML class diagrams is shown in figure 3(b), among which, there are four activity objects e0,ei,ej,en and four relation objects finishToStart.

![Diagram of concurrency block](a)

**Figure 3.** The object-oriented description of process concurrency block

In the object-oriented description of process concurrency block, there is a finishToStart association relation between e0 and ei, so does between e0 and ej, ei and ej, and ej and en. with the above method, the following constraints written in OCL are established.

\[
\text{e_i.linkToPredecessor.kind==WorkSequenceKind. finishToStart and e_j.linkToPredecessor.kind==WorkSequenceKind.finishToStart implies e_i.linkToPredecessor.activityId== e_j.linkToPredecessor. activityId}
\]

3.3 Selection block object-oriented description
In the literature[4], the process selection block consists of two activities e1,e2, and the process model described by Petri nets is shown in figure 4(a). Accordingly object-oriented description described by UML class diagrams is shown in figure 4(b), among which, there are two activity objects e1,e2 and not any relation object.
Figure 4. The object-oriented description of process selection block

In this structure, between $e_j$ and $e_i$ is a kind of exclusive selection relationship which can be used symbol "or" to indicate. In fact, the constraint relation with the use of OCL expression is more accurate. Using the OCL invariant is defined as follows.

context: selectBlock
inv: self.activities->as Set()->size()==1

3.4 Iteration block object-oriented description

In the literature [4], the process iteration block consists of four activities $e_0, e_i, e_j, e_n$ and the process model described by Petri nets is shown in figure 5(a). Accordingly object-oriented description described by UML class diagrams is shown in figure 5(b), among which, there are four activity objects $e_0, e_i, e_j, e_n$ and four relation object finishToStart.
In the object-oriented description of process concurrency block, there is finishToStart association relation between \( e_0 \) and \( e_j \), so does between \( e_i \) and \( e_j \), \( e_i \) and \( e_n \). With the above method, the following constraints written in OCL are established.

\[ e_i.\text{linkToSuccessor.kind==WorkSequenceKind.} \text{finishToStart} \text{ and } e_0.\text{linkToSuccessor.kind==WorkSequenceKind.} \text{finishToStart} \text{ implies } e_i.\text{linkToSuccessor.activityId} == e_0.\text{linkToSuccessor.activityId} \]

So far, according to concepts in SPEM and by using executable UML and OCL, the object-oriented description of four important basic process blocks defined in [4] is achieved.

4 Conclusion

This paper provides a method to convert EPMM elements and process blocks into object-oriented description according to concepts in SPEM and by using executable UML and OCL, attempts to provide a new way of thinking for the future research and wider applications of the software process modelling.

However, the process models based on the object-oriented description can’t completely replace the process models based on Petri net. The reasons are as follows: First, the process model using UML class diagram and object diagram mainly reflects the process static characteristics; Second, UML and OCL are not easy to describe the dynamic behaviour of the process model, such as concurrent, iterative and interactive; Third, UML and OCL are not completely formal languages. Therefore, constructing process models based on the object-oriented description have a wider range of tools to support. Nevertheless constructing process models based on Petri net are is also necessary.

Acknowledgements

This work was financially supported by the Science Foundation of Key Laboratory in Software Engineering under Grant No. 2012SE402, by the Science Foundation of Key Laboratory of IOT Application Technology of Universities in Yunnan Province Grant No. 2015IOT08.
References
[1] Hsueh NL, Shen WH and Yang ZW. Applying UML and software simulation for process definition, verification, and validation. Information and Software Technology, 2008, 50:897-911.
[2] Lee S, Shim J and Wu C. A meta model approach using UML for task assignment policy in software process. Proceedings of the 9th Asia-Pacific Software Engineering Conference on APSEC 2002. IEEE Computer Society, 2002: 376-382.
[3] "Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0", Tech. Rep., Object Management Group., 2008
[4] Li T. Software Evolution Process Meta-Model EPMM[M]. Springer Berlin Heidelberg, 2008.
[5] PAN Wen-lin, LIU Da-xin."Abstract Syntax of Object Role Modelling ". 2010 International Conference on Software and Computing Technology(ICSCT2010), Kunming, China, 2010. IEEE: 25 — 29P.