An Approach Implementing Template-Based Process Development on BPMN

Xiaofeng Cui
Software Engineering Group, BACC
Beijing, China
cuixf@pku.org.cn

***Abstract*—Process development and maintenance are fundamental jobs in process-centered engineering, e.g., software process improvement. A lot of methods have been proposed for process modeling, and many of them have dedicated to improve process reuse by means of pattern or template techniques. In this paper, we propose an approach implementing the definition of Process Template Models and Template Customization Models, as well as automated transformation from these two kinds of models to Process Instance Models, therefore facilitating the reuse-based process development. All the models are defined based on the well-accepted Business Process Modeling Notation (BPMN) modeling language. The generated process instances are standard BPMN models that can be manipulated and executed by all BPMN-compatible mechanisms. We have applied the proposed approach for process development and maintenance in our company, and the working efficiency is improved markedly.**

Keywords—process; template; BPMN; model transformation

I. INTRODUCTION

Process development and maintenance are fundamental jobs in process-centered engineering, e.g., business process management (BPM), software process improvement (SPI), etc. A lot of methods have been proposed for process modeling [1, 2], and many of them have been dedicated to improve process reuse and maintenance by means of pattern, template, as well as other similar techniques [3, 4].

Process patterns, templates, and other kinds of reuse-intended artifacts, store fragments of design that can be easily copied and adapted for recurring circumstances. Pattern "*describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*" [5, 6]. A lot of research efforts have focused on the formality of process patterns, but most are theoretical, and according to Fellmann, Koschmider, and Schoknecht [7], "*the state of empirical evidence in respect to proven positive effects using these approaches is largely unclear*."

Template represents "*a thing that is used as a model for producing other similar examples*" (Oxford Dictionary). The process engineering community does not give a consensus and concise definition of process template. In our opinion, template technique cares less on the problem-solution paradigm than pattern, whereas intends to provide more general-purpose artifacts that can be easily copied and adapted for recurring circumstances and similar designs. Process templates normally refer to various kinds of process fragments that can be tailored, adapted, or customized into concrete executable processes. Because it is simpler and contains less semantic constrains, template technique can be adopted more easily than pattern technique in real-world projects.

Based on our industrial experience, the real-world process development and maintenance are nontrivial jobs. Because there are normally a great deal of processes need to be defined for formality, modified for upgrading, and checked for consistency, and there exist similarities here and there among processes, the simple "copy-and-paste" mode of maintenance eventually leads to a mess (if not a disaster). To cope with this problem, a simple and effective reuse mechanism for process development and maintenance is in highly need. Template technique is supposed to be a good candidate for this purpose.

In this paper, we propose an approach that facilitates template-based process development. We implement the definition of Process Template Models and Template Customization Models, as well as automated transformation from these two kinds of models to Process Instance Models. All the models are defined based on the well-accepted Business Process Modeling Notation (BPMN) [8], as it is the standard modeling language in the field of Business Process Management, and supports process definition, configuration, enacting, monitoring, and analysis. The generated process instances are standard BPMN models that can be manipulated and executed by all BPMN-compatible mechanisms. We have applied the proposed approach for process development and maintenance in our company, and the working efficiency is improved markedly.

The rest of this paper is organized as follows. Section II introduces related work. Section III presents the basic concepts in our proposed approach. Section IV describes the modeling of Process Template Models and Template Customization Models. Section V describes the transformation from the Template Models and Customization Models to Process Instance Models. Section VI presents the application experience of the approach in our company. Section VII gives concluding remarks and future work.

978-1-5090-5507-4/17/$31.00 ©2017 IEEE
ICIS 2017, May 24-26, 2017, Wuhan, China

II. Related Work

Software process modeling is a well-known topic in the Software Engineering research that has been studied for decades [9, 10]. Researchers in this area have addressed the problems of describing and using it, focusing on the idea that a software process is a specific kind of software [10]. Borgoñón et al. [2] seek to identify what software process modeling languages have been defined in last decade, the relationships and dependencies among them and, starting from the current state, to define directions for future research. Their review shows the relevant fact that model-based SPMLs (Software Process Modeling Languages) are being considered as a current trend. Bendraou et al. [1] provide an extensive overview of predominant UML-based languages for software process modeling. To support the comparison of these various approaches, they propose a frame gathering a set of requirements for process modeling. They also evaluate the suitability of UML as a process modeling language and highlight its advantages as well as its drawbacks.

Process pattern is an emerging approach for process reuse. The idea behind process patterns is to capture and reuse process knowledge for recurrent development problems. Tran, Coulette, and Dong [11] point out that "*representing process models based on process patterns to explicit process solutions and factor recurrent process constituents is useful for process understanding as well as process modeling*". They emphasize "*adequate formalization of process patterns*", which means: (1) the process pattern concept must be defined as a model element together with suitable modeling constructs to allow an explicit representation of process patterns applications in process models; (2) the proposed constructs should reflect the variety of process patterns and their applications; (3) a formally defined syntax and semantics should be provided, so that the proposed constructs are computable. They present a UML-based process meta-model that allows explicit representation of process patterns in process models.

In the context of software engineering process, many formalisms and languages have been proposed to describe software process patterns. This multiplicity makes the goal of capitalization and/or reuse of process patterns difficult to be achieved. Jlaiel and Ahmed [3] present a comparative study of process pattern formalisms proposed in the literature and addresses some reflections to deal with problems arising from this survey, in order to propose a new framework for process patterns' capitalization and reuse.

Process templates are also considered valuable resources to be retrieved and reused in other projects [12]. Sharma et al. [13] explore the nature of business rule in business and develop a Business Rule meta-model to represent them. Thereafter, A Process Template is derived from Business Rule meta-model. In their work, business process is considered as a control flow over a set of activities and event. Hence, the process template is built over the activities, event and flow of activities. They claim that Process Template will be mapped to high level process model, but remains as future work. Lin and Strasunskas [12] claim that in order to find a desired template for reuse, the semantics of various process templates should be machine-readable and interoperable. However, the heterogeneity of both model representations and modeling languages makes it difficult to reuse the templates. They [12] adopt one of the emerging semantic web techniques – the semantic annotation of process templates – in order to enhance the interoperability for better reuse of process templates.

According to Fellmann, Koschmider, and Schoknecht [7], "*while there are numerous approaches devising methods and models for reuse or design tool support, there is a lack of empirical research to substantiate the positive effects attributed to the approaches and tools; regarding the more general, broad claims on positive effects (e.g. regarding time, effort and quality), it has to be stated that a validation in this respect is almost completely missing*." Fellmann, Koschmider, and Schoknecht [7] therefore fill this gap by systematically analyzing the available publications. Their paper contributes to the understanding of business process model reuse and consequently also to the knowledge base regarding process model reuse.

Our work inherits the spirit of improving process development efficiency via reuse. We focus on a simple and pragmatic mechanism to achieve the goal of reuse in real-world circumstance. We propose a template-based approach and provide a complete solution from template definition to the generation of concrete executable processes.

III. Basic Conceptions in Our Approach

The basic conceptions in our approach are three kinds of process models: Process Template Model (Template Model for short), Template Customization Model (Customization Model for short), and Process Instance Model (Instance Model for short). Two kinds of languages are used to model the process models: BPMN (Business Process Modeling Notation), which is a standard process modeling language, and PTCL (Process Template Customization Language), which is our extension of BPMN to express the Customization Models. Fig. 1 depicts a meta-model of these basic concepts.

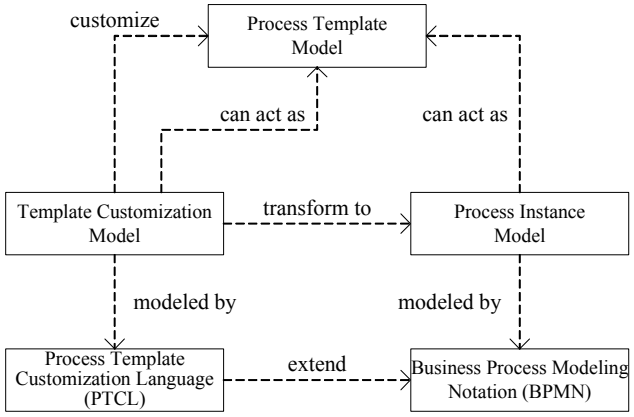


Fig. 1. Basic conceptions of our template-based process models

In our approach, a Template Model is used to model a process template, which represents common features of a set of concrete processes. A Customization Model, on the other hand, expresses the way of building a concrete process based on a Template Model. Finally, the Instance Model is a standard

process model expressed in BPMN, therefore it can be manipulated and executed by any standard BPMN mechanism.

In order to model the Customization Models, we propose PTCL, which is a simple extension to BPMN. PTCL inherits notations of BPMN, while adding three kinds of tags to mark the customization operations on each BPMN element. These customization operations are:

- *add*, which means add one element to the template;
- *delete*, which means delete one element in the template;
- *modify*, which means modify the properties of one element in the template.

Based on one Customization Model and its associated Template Model, one Instance Model can be generated automatically by means of model transformation. The following sections descript the three kinds of models and their transformation in more detail.

IV. Template Models and Customization Models

A. Template Models

According to Fig.1, both Instance Models and Customization Models can act as Template Models. In other words, Template Models can be expressed in BPMN or PTCL.

When a Template Model is expressed in BPMN, it is actually a standard BPMN process model, except that any element or property in the model can be absent or be set a blank value. The absent or blank entities will be set a value when the template is customized. Table I shows an example Template Model that is expressed in BPMN. Fig. 2 shows the diagram of this BPMN model. (In our examples, the process models are extended by a BPMN-compatible process engine, Activiti[1], so that the processes can be executed on the engine.)

TABLE I. Template Model Example

```
<process id="template1" name="template1">
<userTask id="usertask1" name="usertask1"
activiti:candidateGroups="role1"></userTask>
<userTask id="usertask2" name="usertask2"
activiti:candidateGroups="role2">
<extensionElements><activiti:formProperty id="Approve Comment"
type="string" ></activiti:formProperty></extensionElements>
</userTask>
<startEvent id="startevent1" name="Start"></startEvent>
<endEvent id="endevent1" name="End"></endEvent>
<exclusiveGateway id=" exclusivegateway1" name=" Exclusive
Gateway1"></exclusiveGateway>
<sequenceFlow id="flow1" name="" sourceRef="startevent1"
targetRef="usertask1"></sequenceFlow>
<sequenceFlow id="flow2" name="" sourceRef="usertask1"
targetRef="usertask2"></sequenceFlow>
<sequenceFlow id="flow3" name="" sourceRef="usertask2" targetRef="
exclusivegateway1"></sequenceFlow>
<sequenceFlow id="flow4" name="approved" sourceRef="
exclusivegateway1" targetRef="endevent1">
<conditionExpression xsi:type="tFormalExpression">
<![CDATA[${approve == true}]]></conditionExpression>
</sequenceFlow>
<sequenceFlow id="flow5" name="unapproved" sourceRef="
exclusivegateway2" targetRef="usertask1">
  <conditionExpression
xsi:type="tFormalExpression"><![CDATA[${reject ==
true}]]></conditionExpression>
 </sequenceFlow>
</process>
```

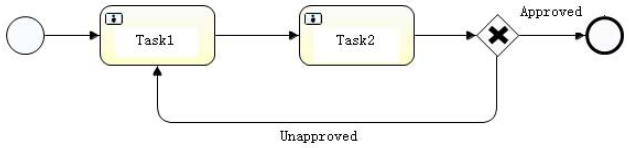


Fig. 2. Process diagram of the Template Model in Table I

When a Template Model is expressed in PTCL, it actually can be transformed to an Instance Model in BPMN in the first place. Then it can be treated in the same way as a BPMN template does. The next section describes the transformation from a Customization Model to an Instance Model.

B. Customization Models

A Template Model can be customized by means of one or more Customization Models. The Customization Model defines the customization entities in the Template Model. The customization operation can be "*add*", "*delete*", or "*modify*".

Table II shows a Customization Model that customizes the Template Model in Table I. Firstly, the Customization Model specifies the template to be customized, i.e., *template1* in this case. Then there are two customization entities that are specified. The first one is *usertask1*, whose name is customized to "*Record Minute*", and the *candidateGroups* property is customized to "*Project Member*". One element "*extensionElements*" is added to *usertask1*, used to define the corresponding fields when this task is executed and the corresponding form is displayed to be filled out. The second customization entity is *usertask2*, whose name is customized to "*Approve Minute*", and the *candiateGroups* property is customized to "*Project Manager*".

TABLE II. Customization Model Example 1

```
<process id="process1" name="process1" template="template1">
<userTask id="usertask1" customization="modify" name="Record
Minute" activiti:candidateGroups="Project Member">
<extensionElements customization="add">
<activiti:formProperty id="1.Conference Date" name="short"
type="string"></activiti:formProperty>
<activiti:formProperty id="2.Conference Venue" name="short"
type="string"></activiti:formProperty>
<activiti:formProperty id="3.Conference Participants"
type="string"></activiti:formProperty>
<activiti:formProperty id="4.Conference Content" name="long"
type="string"></activiti:formProperty>
</extensionElements></userTask>
<userTask id="usertask2" customization="modify" name="Approve
Minute" activiti:candidateGroups="Project Manager"> </userTask>
</process>
```

[1] http://www.activiti.org

Table III shows another Customization Model that customizes the Template Model in Table I. In this Customization Model, two customization entities are specified differently to the last example. The first customization entity is *usertask1*, whose name is customized to "*Record View*", and the *candidateGroups* property is customized to "*Review Secretary*". One element "*extensionElements*" is added to *usertask1*, used to define the corresponding fields when this task is executed and the corresponding form is displayed to be filled out. The second customization entity is *usertask2*, whose name is customized to "*Approve Review*", and *candiateGroups* property is customized to "*Review President*".

TABLE III. CUSTOMIZATION MODEL EXAMPLE 2

```
<process id="process2" name="process2" template="template1">
<userTask id="usertask1" customization="modify" name="Record
Review" activiti:candidateGroups="Review Secretary">
<extensionElements customization="add">
<activiti:formProperty id="1.Review Type" type="enum">
<activiti:value id="Phase Review"></activiti:value>
<activiti:value id="Milestone Review"></activiti:value>
<activiti:value id="Acceptance Review"></activiti:value>
</activiti:formProperty>
<activiti:formProperty id="2.Review Date" type="string">
</activiti:formProperty>
<activiti:formProperty id="3.Review Venue" type="string">
</activiti:formProperty>
<activiti:formProperty id="4.Review Board" type="string">
</activiti:formProperty>
<activiti:formProperty id="5.Review Artifacts" type="string">
</activiti:formProperty>
<activiti:formProperty id="6.Review Result" type="enum">
<activiti:value id="Pass without problem"></activiti:value>
<activiti:value id="Pass with problem"></activiti:value>
<activiti:value id="Not pass"></activiti:value></activiti:formProperty>
</extensionElements></userTask>
<userTask id="usertask2" customization="modify" name="Approve
Review" activiti:candidateGroups="Review President"></userTask>
</process>
```

V. GENERATION OF PROCESS INSTANCE MODELS

In our approach, every Customization Model can be transformed to an Instance Model, i.e., BPMN model. The transformation program reads the Customization Model and its associated Template Model, then modifies, deletes, or adds elements in the Template Model according to the customization entities in the Customization Model. The result of transformation is a standard BPMN file as the Instance Model. The transformation can also generate the BPMN diagram, by means of the modification on the Template Model's diagram. Fig. 3 depicts the algorithm for generating the Instance Models from the Template Models and Customization Models.

Table IV shows the Instance Model transformed from the Customization Model in Table II. The Customization entities in the Customization Model are transformed to the concrete elements in the resultant Instance Model. Fig. 4 shows the diagram of the Instance Model. Fig. 5 shows the form displayed on the first step (*usertask1*, i.e., "*Record Minute*") when this process is executed.

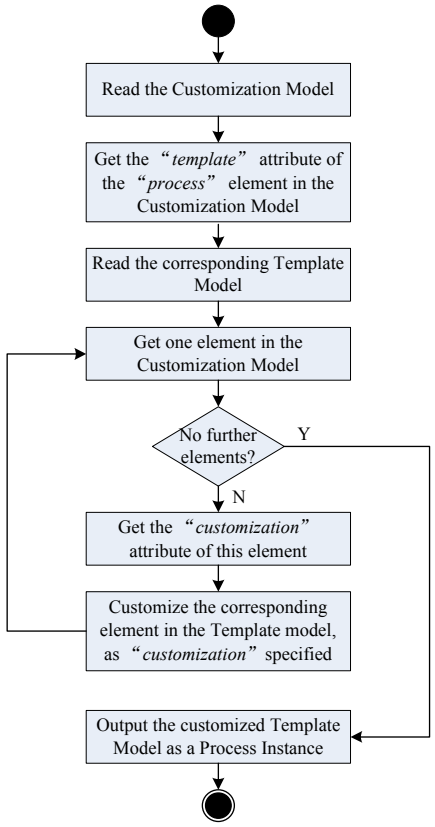


Fig. 3. The algorithm for generating the Instance Models

TABLE IV. THE GENERATED PROCESS INSTANCE 1

```
<process id="template1" name="template1">
<userTask id="usertask1" name="Record Minute"
activiti:candidateGroups="Project Member">
<extensionElements><activiti:formProperty id="1.Conference Date"
name="short" type="string"></activiti:formProperty>
<activiti:formProperty id="2.Conference Venue" name="short"
type="string"></activiti:formProperty>
<activiti:formProperty id="3.Conference Participants"
type="string"></activiti:formProperty>
<activiti:formProperty id="4.Conference Content" name="long"
type="string"></activiti:formProperty></extensionElements>
</userTask>
<userTask id="usertask2" name="Approve Minute"
activiti:candidateGroups="Project Manager">
<extensionElements>
<activiti:formProperty id="Approve Comment" type="string" >
</activiti:formProperty></extensionElements></userTask>
<!-- the same as Table I from here down -->
```

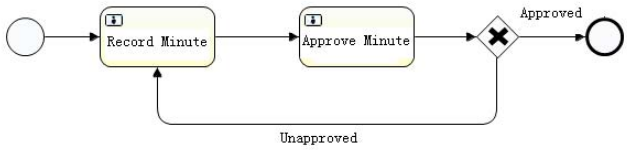


Fig. 4. The generated diagram of process instance 1

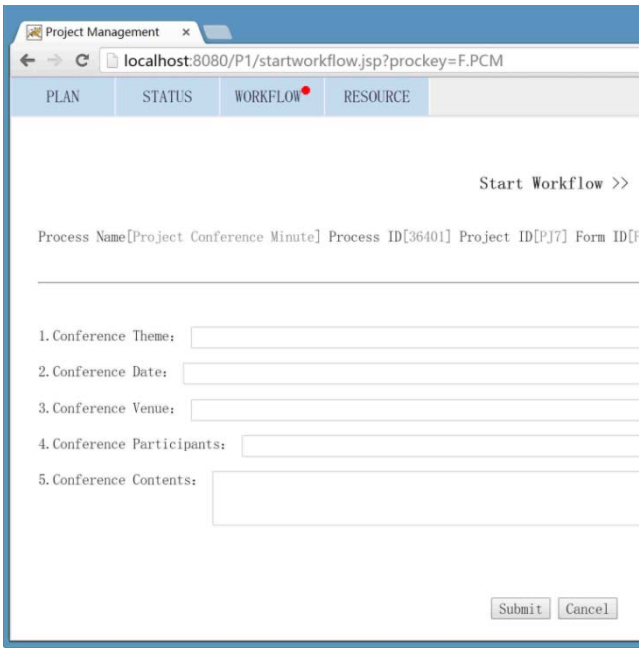


Fig. 5. The generated form of 'Record Minute' task in process instance 1

Table V shows the Instance Model transformed from the Customization Model in Table III. The customization entities in the Customization Model are transformed to the concrete elements in the resultant Instance Model. Fig. 6 shows the diagram of the Instance Model. Fig. 7 shows the form displayed on the first step (*usertask1*, i.e., "*Record Review*") when this process is executed.

TABLE V. THE GENERATED PROCESS INSTANCE 2

```
<process id="template1" name="template1">
<userTask id="usertask1" name="Record Review"
activiti:candidateGroups="Project Manager"><extensionElements>
<activiti:formProperty id="1.Review Type" type="enum">
<activiti:value id="Phase Review"></activiti:value>
<activiti:value id="Milestone Review"></activiti:value>
<activiti:value id="Acceptance Review"></activiti:value>
</activiti:formProperty>
<activiti:formProperty id="2.Review Date" type="string">
</activiti:formProperty>
<activiti:formProperty id="3.Review Venue" type="string">
</activiti:formProperty>
<activiti:formProperty id="4.Review Board" type="string">
</activiti:formProperty>
<activiti:formProperty id="5.Review Artifacts" type="string">
</activiti:formProperty>
<activiti:formProperty id="6.Review Result" type="enum">
<activiti:value id="Pass without problem"></activiti:value>
<activiti:value id="Pass with problem"></activiti:value>
<activiti:value id="Not pass"></activiti:value>
</activiti:formProperty></extensionElements></userTask>
<userTask id="usertask2" name="Approve Review"
activiti:candidateGroups="Review President">
<extensionElements><activiti:formProperty id="Approve Comment"
type="string" ></activiti:formProperty></extensionElements></userTask>
<!-- the same as Table I from here down -->
```

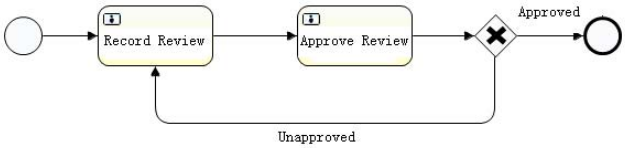


Fig. 6. The generaed diagram of process insance 2

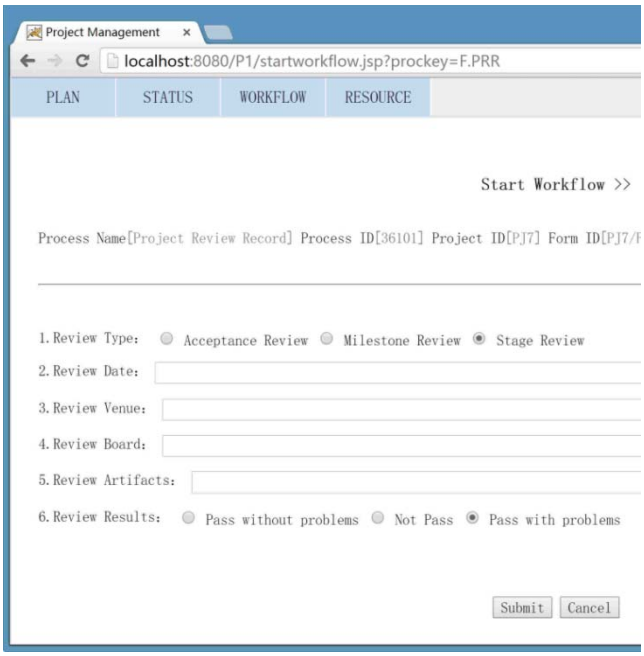


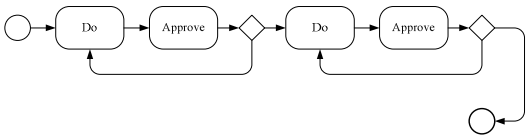
Fig. 7. The generated form of 'Record Review' task in process instance 2

VI. APPLICATION EXPERIENCE OF THE APPROACH

We have used the proposed approach in our company for software processes development and maintenance for years. These processes cover project management, configuration management, quality assurance, and organizational process management areas. Many of them have common characteristics, e.g., similar structures, even for the processes in the different areas. Table VI lists one of our defined Template Models and it related Customization Models for the purpose of reuse.

As an example, the *DADA.bpmn* Template Model abstracts the common characteristics of a group of processes that have a "*Do-Approve-Do-Approve*" structure (four major steps in the process). The "*Project Problem Report"* process is a process instance that has this structure, where the "*Do-Approve-Do-Approve*" structure is specialized to "*Report-Approve-Correct-Approve*" steps of the process. The "*Requirements Change Report"* process is another process instance of this kind, where the "*Do-Approve-Do-Approve*" structure is specialized to "*Report-Approve-Change-Approve*" steps of the process. Therefore these process instances that share the common characteristics can be defined by means of the different Customization Models based on one same Template Model. Then the Instance Models can be generated automatically from the Customization Models.

TABLE VI. SOME TEMPLATE AND CUSTOMIZATION MODELS IN OUR REAL-WORLD APPLICATIONS

Template Models (Characteristics)	Template Model Diagrams	Customization Models	Instance Model Names (Characteristics)
DADA.bpmn (Do-Approve-Do-Approve)		PPR.PTCL	Project Problem Report (Report-Approve-Correct-Approve)
		RCR.PTCL	Requirements Change Report (Report-Approve-Change-Approve)
		OPAP.PTCL	Organization Process Appraisal Report (Plan-Approve-Appraisal-Approve)
		OPAC.PTCL	Organization Process Action Report (Plan-Approve-Action-Approve)
		CRMO.PTCL	Organization Configuration Repository Maintenance Report (Plan-Approve-Maintenance-Approve)
		CRMP.PTCL	Project Configuration Repository Maintenance Report (Plan-Approve-Maintenance-Approve)

The list in Table VI shows that in real-world applications, there are normally a lot of processes with similar structures and characteristics. By means of the proposed template-based process definition, all common structures and characteristics can be modeled with a little number of Template Models, while the concrete processes can be defined by the customization of Template Models. The Template Models and Customization Models are much simpler to be defined and maintained than a lot of cloned processes instances, because Template Models only define common characteristics, and Customization Models only define customization entities. There are no duplicated elements among all the models.

In our company, by means of the proposed approach, the number of processes need to be maintained is decreased, and the consistency between the similar processes can be guaranteed easily. As a result, the efficiency of process development and maintenance is improved markedly.

VII. CONCLUSIONS AND FUTURE WORK

We implement a template-based approach for process development. The proposed Template Models and Customization Models are expressed with the standard BPMN language and its simple extension (the proposed PTCL language). The Instance Models then can be generated by means of model transformation from the Customization Models. The proposed approach facilitates process model reuse, therefore the job of process development and maintenance can be much more efficient. All the proposed models are based on the standard BPMN language, so that the models can be understood easily and manipulated by various BPMN-compatible mechanisms.

We have developed the automated transformation program to generate Instance Models based on the specified Customization Models and Template Models. The Customization Models are currently specified by means of plain text editor. We are planning to develop a graphical modeling interface, so that the Customization Models can be specified interactively and displayed graphically.

REFERENCES

[1] R. Bendraou, J. M. Jézéquel, M. P. Gervais, and X. Blanc, "A Comparison of Six UML-Based Languages for Software Process Modeling," IEEE Transactions on Software Engineering, vol. 36, pp. 662-675, 2010.

[2] L. García-Borgoñón, M. A. Barcelona, J. A. García-García, M. Alba, and M. J. Escalona, "Software process modeling languages: A systematic literature review," Information & Software Technology, vol. 56, pp. 103-116, 2014.

[3] N. Jlaiel# and M. B. Ahmed#, "Reflections on How to Improve Software Process Patterns Capitalization and Reuse," in Information and Knowledge Engineering, 2010, pp. 30-35.

[4] L. Aldin and S. De Cesare, "A literature review on business process modelling: new frontiers of reusability," Enterprise Information Systems, vol. 5, pp. 359-383, 2011.

[5] C. Alexander, S. Ishikawa, M. Silverstein, M. ~Jacobson, and I. ~Fiksdahl-King, "A Pattern Language," Oxford University Press, 1977.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software: Addison-Wesley Longman Publishing Co., Inc., 1995.

[7] M. Fellmann, A. Koschmider, and A. Schoknecht, "Analysis of Business Process Model Reuse Literature: Are Research Concepts Empirically Validated?," Lecture Notes in Informatics, vol. P-225, pp. 185-192, 2014.

[8] OMG, "Business Process Model and Notation (BPMN) Version 2.0," ed, 2011.

[9] H. Krasner, J. Terrel, A. Linehan, P. Arnold, and W. H. Ett, "Lessons Learned from a Software Process Modeling System," Communications of the Acm, vol. 35, pp. 91-100, 1992.

[10] L. Osterweil, "Software processes are software too," in Proceedings of the 9th international conference on Software Engineering, 1987.

[11] H. N. Tran, B. Coulette, and B. T. Dong, "Modeling Process Patterns and Their Application," in Proceedings of the International Conference on Software Engineering Advances, 2007, p. 15.

[12] Y. Lin and D. Strasunskas, "Ontology-based semantic annotation of process templates for reuse," presented at the Proceeding of 10th CAiSE/IFIP8.1/EUNO, International Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD), Porta, Portugal, 2005.

[13] D. K. Sharma, N. Prakash, H. Sharma, and D. Singh, "Automatic construction of process template from business rule," in Contemporary Computing (IC3), 2014 Seventh International Conference on, 2014, pp. 419-424.