Verification of Invariant Properties of Business Process based on Formal Approach

Shimpei Sasaki
Graduate School of Decision Science and Technology
Tokyo Institute of Technology
Tokyo, Japan
sasaki.s.ag@m.titech.ac.jp

Junichi Iijima
Graduate School of Decision Science and Technology
Tokyo Institute of Technology
Tokyo, Japan
iijima.j.aa@m.titech.ac.jp

***Abstract*-Recently, the concept of business process management (BPM) is in the spotlight and it is getting popular to design, enact and monitor a business process from the viewpoint of BPM. It often happens, however, that designed business process models often do not satisfy correctness properties such as executability, satisfiability and so on. Therefore it is necessary to check those correctness properties of a business process at its design phase.**

Although most of BPM systems have a functionality of simulation, it is not based on rigorous mathematical background. In this paper, we propose an approach to use formal specification in order to verify invariant properties of a business process rigorously at the design phase. Toward this goal, firstly, we define the transformation from a business process model in XML Process Definition Language (XPDL), which is one of the standard description of a process in BPM, to the model in VDM specification language (VDM-SL), which is one of popular formal specification language for software. Then we verify the invariant properties of the transformed model with VDM development support tool called VDMTools.

Keywords-XPDL; VDM-SL; VDMTools; formalization; verification; invariant

I. INTRODUCTION

Recently, the concept of business process management (BPM) is getting popular in business context. One of characteristics of BPM is to repeat the cycle which consists of process design, system configuration, process enactment and diagnosis, and improve the target business process continuously.

Although there are many opportunities to design business processes, the models of those business processes often do not satisfy correctness properties such as invariants, executability, satisfiability and so on.

Verification of a business process at the design phase has a great advantage; for example, it will shorten time to modify the model, reduce costs and utilize resources more properly if correctness properties of the business process model are verified.

Although most of BPM systems have a functionality of simulation, it is not assured of the correctness of the designed model for it is not based on rigorous mathematical background.

In this paper, we propose a formal approach for assuring invariant properties of a business process at the design phase. Concretely speaking, we adopt a business process model described in XPDL and convert it into a formal specification described in VDM-SL. VDM-SL is one of the well-known formal specification language used in software development. One of the distinguished characteristics of VDM-SL is that we can discuss on the invariant properties of the described system using the VDM development support tool called VDM Tools.

This paper consists of five sections. In section 2, we introduce several basic concepts including XPDL and VDM, and survey researches related to formal approach in BPM. Next, we define a formal structure of a business process model described in XPDL and VDM-SL as well. We also define transformation of a model in XPDL into that in VDM-SL. In section 4, we discuss on implementation of the transformation tool developed in this research and show how to verify the invariant properties of the target business process using VDM-Tools. Finally, we conclude our research and discuss our future work in section 5.

II. BACKGROUND KNOWLEDGE

XPDL is an XML-based process definition language initiated by WfMC (Workflow Management Coalition). Since XPDL is based on XML syntax, it enables us to interchange data and integrate web services easily. However, there is a difficulty to understand a model described in XPDL because of its XML-based description. Therefore, the graphical language, BPMN (Business Process Modeling Notation), will be used in this paper when we show a business process model.

VDM (Vienna Development Method) is a model-oriented formal method used in systems development. It was developed in IBM Vienna Laboratory in the mid 1970s. The specification language of VDM is called VDM-SL (VDM Specification Language). There is an OO-extension called VDM++. In this paper, VDM-SL will be used to model business processes.

Since a model described in VDM-SL has two characteristics, that is, abstractness and rigorousness [1], it can be verified readily and rigorously.

So far there are several formal approaches in BPM. One of the representatives is an approach based on Petri net. Although

1-4244-1312-5/07/$25.00 © 2007 IEEE

approaches based on logical languages are another representative group [5], [8], it is difficult for people to take the approach if he/she does not have enough background knowledge.

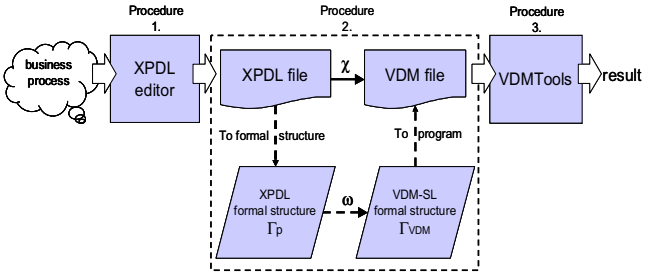

Figure 1. Outline

Li and Iijima proposed an approach to check correctness of a business process described in XPDL after converting it into XSSL-based expression [8], where XSSL proposed in the paper is a newly developed language based on Situation calculus in XML syntax.

In this paper, we focus on to assure invariant properties of a business process and propose a new approach to verify the correctness of the business process rigorously with XPDL and VDM-SL.

III. Business Process Modeling and Transformation

A. Outline

The outline of verification process of a business process is shown in Fig. 1.

Step 1: Design a business process model in XPDL.

In this paper, some rules to design the model in XPDL are assumed to be satisfied. Together Workflow Editor Community Edition 2.0-2 is used as an XPDL editor in the above figure.

Step 2: Transform from the model described in XPDL to the model in VDM-SL.

In Figure 1, there are two transformations; χ and ω, informal one and formal one, respectively, where χ is based on ω. Using the formal structure, it is possible to define the structure of the model and to understand correspondence between the models.

The formal structure in XPDL is represented by γ_p, that in VDM-SL by γ_{VDM} and corresponding sets are denoted by Γ_p and Γ_{VDM}, respectively, where the transformation is implemented in Java.

Step 3: Verify the model with VDM-Tools.

In this paper, we verify the invariant properties of a target business process and executability of some activity sequence.

B. The Model in XPDL and Formalization of It

In this paper, we transform XPDL-based business process into VDM-SL expression and verify the transformed model. Although there are some ambiguous points in XPDL syntax, we introduce the following five rules to design XPDL-based business process model. For the readability, XPDL tags are described in Sans-Serif font in this paper.

Rule 1: The contents of the activities must be described in the element Description of Activity.

Rule 2: The invariants of the parameters must be described in the element Description of DataField after describing "<invariant>::=".

Rule 3: If the parameter is the option type, it is to be described "<option>" in the element Description of DataField.

Rule 4: If both of "<invariant>::=" and "<option>" are to be described, it is necessary to write one blank line between them.

Rule 5: Each of two conditions of transitions from a XOR split is ambivalent.

Concerning Rule 1, 2 and 3, the element Description is not usually used like that, but for notes in natural language.

A model described satisfying the above rules in XPDL is formalized as follows:

Definition 1. *A formal structure of a business process model in XPDL is*

$$\gamma_p = \langle A, Pm, PV, pms, pvs, conseq, inv, init, Tr \rangle$$

, where

A: a set of activities,

Pm: a set of parameters,

PV: a set of the values which the parameter has in the process,

$pms : A \rightarrow \mathbf{P}(Pm)$*, a mapping from an activity to the corresponding parameter set,*

$pvs : Pm \rightarrow PV$*, a mapping from a parameter to the corresponding value set,*

$conseq : A \rightarrow \mathbf{A}(PV)$*, a mapping from an activity to its consequence,*

$inv : Pm \rightarrow \mathbf{A}(PV)$*, a mapping from a parameter to its invariant,*

$init : Pm \rightarrow \bigcup_{p \in Pm} pvs(p)$ *s.t.* $init(p) \in pvs(p)$*, a mapping from a parameter to its initial value,*

$Tr \subset A \times A \times JoT \times \mathbf{A}(PV)$*, where JoT is the set of join types and consists of ANDjoin, XORjoin, and NOTjoin.*

$\mathbf{P}(Pm)$ denotes the power set of Pm, and $\mathbf{A}(PV)$ the set of conjunctive normal forms with the closed literals defined on PV^n for $n \in \mathbf{N}$.

Fig. 2 shows the meta-model of the components of XPDL.

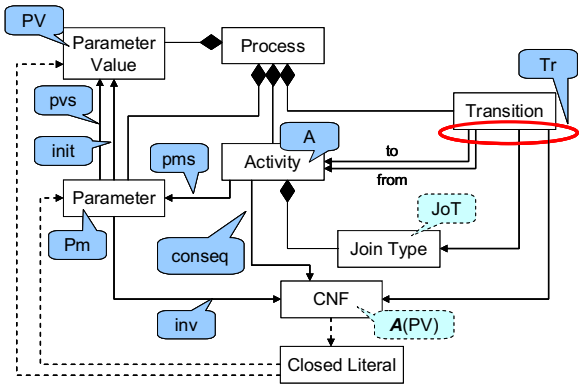

Figure 2. Process meta-model for XPDL

C. The Model in VDM-SL and Formalization of It

A state of a business process is represented by a tuple of values of the variables. Therefore activities of business processes exactly correspond to operations in VDM-SL since they change the values of variables in the model.

Although the order of operations is important, there is no way to represent it explicitly in VDM-SL. Therefore we introduce flag variables in order to show whether an operation is finished or not. Each variable is represented as "(operation name)Status", and has "nil" (not defined) if the operation is not finished and while it is represented <done> if it is finished.

The model in VDM-SL based on the above idea is formalized as follows:

Definition 2. *A formal structure of a business process model in VDM-SL is*

$\gamma_{VDM} = <O, Var, VV, vars, vvs, inv, init, pre, post>$

, where

O: a set of operations,

Var: a set of variables,

VV: a set of the values which each variable has in the process,

vars : $O \rightarrow \boldsymbol{P}(Var)$, *a mapping from an activity to the corresponding variable set,*

vvs : $Var \rightarrow VV$, *a mapping from a variable to the corresponding value set,*

inv : $Var \rightarrow \boldsymbol{A}(VV)$, *a mapping from a variable to its invariant,*

init : $Var \rightarrow \bigcup_{v \in Var} vvs(v)$ *s.t.* $init(v) \in vvs(v)$, *a mapping from a variable to its initial value,*

pre : $O \rightarrow \boldsymbol{A}(VV)$, *a mapping from an operation to its pre-condition,*

post : $O \rightarrow \boldsymbol{A}(VV)$, *a mapping from an operation to its post-condition.*

$\boldsymbol{P}(Var)$ denotes the power set of *Var*, and $\boldsymbol{A}(VV)$ the set of the conjunctive normal forms with the closed literals defined on VV^n for $n \in \boldsymbol{N}$.

D. Transformation from the model in XPDL to the model in VDM-SL

According to Definition 1 and 2, the formal transformation from a model in XPDL to its corresponding model in VDM-SL is defined as follows:

Definition 3. *A formal structure of a business process model in XPDL*

$\gamma_p = <A, Pm, PV, pms, pvs, conseq, inv, init, Tr>$

is transformed by the mapping ω *by*

$\omega(\gamma_p) = <A, Var, VV, vars, vvs, inv, init, pre, post>$

, where

$$Var = Pm \cup \{status_i\}\ (i \in A),$$

$$VV = PV \cup \{\{done, \Delta\}\},$$

$$\forall i, i \in A, \begin{cases} vars(i) = pms(i) \cup \{status_i\}, vvs(status_i) = \{done, \Delta\} \\ inv\{status_i\} = \{\}, init\{status_i\} = \Delta \end{cases}$$

post is defined with $a \in A$ *as follows:*

$$post(a) = status_a ET(done) \wedge conseq(a)$$

pre is defined with $b \in A$ *as follows:*

(1) If there is/are $a \in A$ *such that* $(a, b, jot, cond_{ab}) \in Tr$,

$$pre(b) = status_b ET(\Delta) \wedge preJOIN(b)$$

, where

$preJOIN(b)$

$$= \begin{cases} status_a ET(done) \wedge cond_{ab}\ (if\ jot = NOTjoin) \\ \wedge_i \{status_{a_i} ET(done) \wedge cond_{a_i b}\}\ (if\ jot = ANDjoin) \\ \oplus_i \{status_{a_i} ET(done) \wedge cond_{a_i b}\}\ (if\ jot = XORjoin) \end{cases}$$

$\oplus$ *represents exclusive OR*

(2) If there is no $a \in A$ *such that* $(a, b, jot, cond_{ab}) \in Tr$,

$$pre(b) = \{status_b ET(\Delta)\}$$

Δ denotes undefined, and $status_i ET(*)$ is the predicate representing that the flag variable of the operation *i* is equal to *. {} represents that there is no literal. In order for readability, some parts are not described in CNF.

IV. Implementation of Transformation Tool and Verification of the Model

A. Implementation of transformation tool

Based on the mapping defined in 3.4, the transformation tool is implemented in Java.

After reading the XPDL file, the tool checks every node and puts the values of the nodes in the variables with DOM. Then, the values are arranged to the VDM-SL expression, and output into a new file.

B. Contents of verification

In this paper, the contents of the verification are two points; the executability of an activity sequence and the satisfaction of the invariants.

Executability of an activity sequence:

First, the executability of an activity sequence is defined as follows:

An activity sequence $a_1, a_2, ..., a_n$ is executable.
$\Leftrightarrow$ *An activity sequence $a_1, a_2, ..., a_n$ satisfies the following:*

1. *for every $i \in \{1, 2, ..., n-1\}$,*
   the contents of $a_1, a_2, ..., a_i$ is finished.
   $\Rightarrow$ *the pre-condition of a_{i+1} is satisfied.*
2. *the initial state satisfies the pre-condition of a_1 where the initial state is the set of the initial values of all variables in the process.*

Satisfaction of the invariants:

The invariants of the process will be checked for each case. This is a great advantage of using VDM.

Checking these two points leads to the verification of correctness of the model.

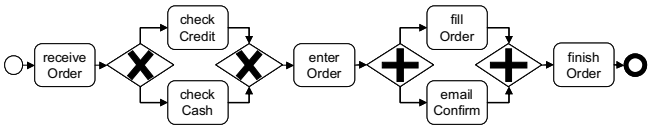

Figure 3. Order process (BPMN)

C. *Verification of a case*

In order to illustrate our approach described in the previous sections, let us take an order process shown in Fig. 3 as an example.

Verification of executability of an activity sequence:

Suppose that we would like to check whether an activity sequence "receive order", "check cash", "enter order", "fill order" and "finish order" with order one by cash is executable or not. In this case, we can do it if we input "`print receiveOrder(1, <cash>), checkCash(), enterOrder(), fillOrder(), emailConfirm()`" into VDMTools. An activity sequence is executable if it outputs no error message.

Verification of satisfaction of the invariants:

Suppose that we would like to check whether the invariant of the parameter cardNumber, that is, the condition that cardNumber should be three digits if in case that the quantity is 1, the way to pay is by credit card and the card number is 1234.

In this case, it outputs the following error message:

```
(no return value)
C:/order.vdm, l. 17, c. 61:
 Run-Time Error 99: State invariant was broken
C:/order.vdm, l. 59, c. 1:
 Run-Time Error 58: The pre-condition evaluated to false
…
```

if we input "`print receiveOrder(1, <credit>), checkCredit(), enterOrder(), fillOrder(), emailConfirm(), finishOrder()`" into VDMTools. The output message shows that receiveOrder was normally finished, but pre-condition of the following activities evaluated false for the invariant of the parameter cardNumber was broken.

V. CONCLUSION

In this paper, we propose a formal approach to verify correctness of a business process, especially, invariants of the process. Firstly, models in XPDL and in VDM-SL are formally defined and then the transformation is also defined in a formal way. The transformation tool was implemented in Java. Next, we illustrate how to verify correctness of the transformed model with VDMTools, especially with respect to two points; executability of an activity sequence and satisfaction of the invariants of a process of an activity sequence.

REFERENCES

[1] J.Fitzgerald and P.G.Larsen, *Modelling Systems, Practical Tools and Techniques in Software Development*, Cambridge University Press, 1998.
[2] CSKsystems, *VDMTools VDM-SL Toolbox User Manual*, 2006.
[3] VDM INFORMATION WEB SITE, http://www.vdmtools.jp/
[4] W.M.P van der Aalst, A.H.M. ter Hofstede, and Mathias Weske, "Business Process Management: A Survey", *BPM 2003*, LNCS 2678, Springer, 2003, pp. 1-12.
[5] M. Havey, *Essential Business Process Modeling*, O'Reilly, August 2005.
[6] WfMC, *Process Definition Interface - XML Process Definition Language*, Document Number WFMC-TC-1025 Document Status - Final Version 2.00, 2005.
[7] W.M.P van der Aalst, "Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language", *QUT Technical Report*, FIT-TR-2003-06, Queensland, Australia, 2003.
[8] B. Li and J.Iijima, "Formal Verification of XPDL-based Business Process Definition", *International Journal of Business Process Integration and Management*, 2007., unpublished.