DERIVING CANONICAL BUSINESS OBJECT OPERATION NETS FROM PROCESS MODELS

Wang Zhaoxia[1,2,3], Wang Jianmin[2], Wen Lijie[2] and Liu Yingbo[2]
[1]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[2]*School of Software, Tsinghua University, Beijing, China*
[3]*Department of Logistical Information Engineering, Logistical Engineering University, Chongqing, China*

Keywords: Process model, Business object, Object life cycle, Workflow Coloured Petri Net, Operation net.

Abstract: Process model defines the business object operation orders. It is necessary to validate that the business object operation sequences are consistent with the reference business object life cycle. In this paper we propose an approach for deriving the canonical business object operation nets from process models which are modelled with workflow coloured Petri net, i.e. WFCP-net. Our approach consists of 3 steps. First, the tasks, which access a certain business object, are modelled with task operation nets. Second, the WFCP-net is rewritten with these task operation nets. Third, the business object operation net is reduced to the canonical one.

1 INTRODUCTION

Generally, the term *business object* denotes data which is handled in a business process (Engels,2001; Ryndina 2006). Life cycle of a particular business object describes its whole life process from creation to death. It provides the object's state constraint criteria which define the state pre-condition for an operation on the object and the transition rule when the operation is handled. Thus, the manipulating of the object in a business process should conform to the state constraint criteria of the object. If the execution of a task with relevant operations on the object violates the criteria, it will cause undesired results, e.g., exceptions, halting, etc. In order to avoid the above undesired result, a mechanism is necessary to validate whether the operations on the object in the process model complies with the state constraint criteria of the object. To some extent, it is a problem of model checking (Clarke, E. et.al, 1996) which includes three parts: system specification, system model and checking algorithm. The reference life cycle of the business object is regard as system specification while the operation diagram of the object in the process is regard as system model. The validating algorithm is equal to checking algorithm. In the rest of this paper, we use term *business object* or *object* to express the *data* which is handled in a business process.

A key difficulty of the above problem is how to derive the object operation diagram from a process. Currently some researches are to capture the flow of objects in a process and to describe an object's change between tasks. However, these researches cannot differentiate between the tasks in the process and the operations on the object. A method to capture the flow of object is to annotate task with *read*/*write* operation in the process model (Sun et al., 2006; Lee et al., 2007). However, abstract *read* and *write* operations are not enough to represent the abundant concrete operations (such as *check in*, *check out*, and *release*). Other approaches (Engels, 2001; Ryndina 2006; Küster et al., 2007) focus on the one-to-one relationship between the tasks and the operations. But in some application domain, e.g., product lifecycle management widely used in manufacturing enterprise, it is not the one-to-one relationship between the tasks and the operations. In one word, current process model does not describe the concrete operation information of objects. Motivated by the difficulty, we propose an approach to derive the canonical business object operation net from a process model by refinement method and reduction rules.

The remainder of this paper is organized as follows. Section 2 presents a real-life example which

Zhaoxia W., Jianmin W., Lijie W. and Yingbo L. (2009).
DERIVING CANONICAL BUSINESS OBJECT OPERATION NETS FROM PROCESS MODELS.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 182-187
DOI: 10.5220/0001996801820187

contains a reference object life cycle and a business process to illustrate the idea in this paper. In Section 3 relevant concepts are introduced. Section 4 proposes an approach to derive a canonical object operation net from a process model. Section 5 discusses related work. Finally, conclusions and future work are shown in Section 6.

2 AN EXAMPLE

In this section, we give an example from TiPLM system (TiPLM, 2008). TiPLM system is a kind of PLM (Product Lifecycle Management) system which has two main functionalities: the product structure management and the workflow management.

2.1 A Reference Object Life Cycle

A business object has a life cycle through which it evolves and consequently transfers among various states. The standard life cycle of the engineering data in TiPLM system is shown in Figure 1. States are indicated by circles (double circle denotes the *accept state*). Transition between states is indicated by directed arc labelled with operation. For instance, if an object is in the *state of being checked out*, the operation *check in* is enabled. When the operation is completed, a state transition from the *state of being checked out* to the *state of being checked in* will be triggered.

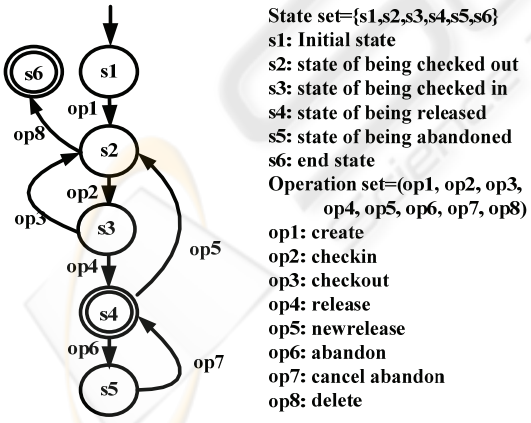

Figure 1: A reference object life cyclefor engineering data.

2.2 A Business Process Example

Figure 2 illustrates a design and review process of engineering drawings in TiPLM system. The key steps of this process include task *designing*, task *verifying*, task *reviewing*, task *approving* and task *releasing*. After the engineering drawings are designed, they will be checked step by step. Once the checking of them is failed, they will be returned to task *designing* for modification.

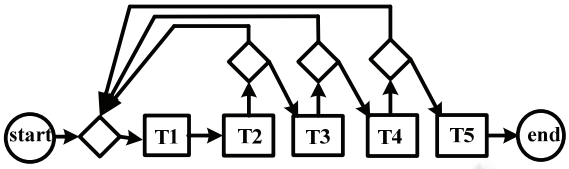

Figure 2: Design and review process ofengineering drawings.

For instance, the operation information on the object of task *designing* is:

On the one hand, if the object is not existed, the operation *create* is executed. When the object is created, its current state is the *state of being checked out*. Then under the condition that the state of the object is the *state of being checked out*, designer generates the 2D drawings or 3D models. Once the work is completed, the operation *check in* is triggered and the object is checked into database. Then, the state of the object will change to the *state of being checked in*. On the other hand, if the design is returned by the some subsequent task, such as task *verifying*, *reviewing* and *approving*, the operation *check out* is executed and the object's state changes from the *state of being checked in* to the *state of being checked out*. Then the designer can verify the object.

Due to the limit of space, we omit the description of other tasks. The relationship between tasks in the process model and operations on the object is shown in Table 1.

Table 1: Tasks in the process and operations on the object.

task	relevant operations on the object
designing	create, check in, check out
verifying	null
reviewing	null
approving	null
releasing	release

It is apparent that there is no one-to-one corresponding relation between the tasks and the operations. Therefore, in order to identify whether the operation sequence of an object in a business process is consistent with the reference object life cycle, a mechanism to characterize the operations on the object in the process model and to derive the

object operation net is needed. The concrete solution is elaborated in Section 4.

3 PRELIMINARIES

This section introduces the key concepts relevant with our work.

3.1 Object Life Cycle

Generally, a non-deterministic finite state machine is a common means of modelling object life cycle (Stumpiner, M.& Schrefl, M., 2000). The formal definition of object life cycle is as follows.

Definition 3.1 (Object Life Cycle). Given an object O, its object life cycle $OLCo = (T, S, \delta, s_\alpha, S_\Omega)$ where:

$-T$ is a finite set of operations,

$-S$ is a finite set of states,

$-\delta: S \times T \to 2^S$ is the transition function, mapping each state and operation to a set of possible successor states,

$-s_\alpha \in S$ is the initial state, and $S_\Omega \subseteq S$ is a set of *accept states*.

For example, the reference object life cycle of business object shown in Figure 1 can be described as:

$T = \{op1, op2, op3, op4, op5, op6, op7, op8\}$,

$S = \{s1,\ s2, s3,\ s4,\ s5, s6\}$, $s_\alpha = s1$, $S_\Omega = \{s4, s6\}$.

3.2 WFCP-net

As a kind of workflow net, WFCP-net is extended based on WF-net (Aalst, 1998) and Coloured Petri Nets (Jensen, 1997). For a detailed introduction to WFCP-net, the reader is referred to (Liu et al., 2002).

Definition 3.2 (WFCP-net). A $CPN =< \Sigma, S, T, F, C, G, E, I >$ is a WFCP-net if and only if:

(i) IN and OUT are subsets of S; IN which only has one element is a set of workflow start places, OUT which may have one or many elements is a set of terminal places formal description: $IN, OUT \in S$: $[|IN| = 1; |OUT| \geq 1]$ and $\forall\, i \in IN, \cdot i = \emptyset; \forall\, o \in OUT, o \cdot = \emptyset$.

(ii) $\forall\, x \in S \cup T \wedge x \notin IN \wedge x \notin OUT$, x is on the path from $i \in IN$ to $o \in OUT$.

(iii) Tp is category of transition T:

$\forall t \in T: Type\big(Tp(t)\big) \in TT$,

$TT = \{Auto, User, Event, Time, Dumb\}$.

(iv) The initialization function: $I = \begin{cases} C(i)_{MS}, & p = i \\ \emptyset, & p \neq i \end{cases}$.

To guarantee the process is started and ended correctly, we add two types of task: t^{start} and t^{end}. For instance, the model of the business process example in Section 2 is shown in Figure 3(a).

4 DERIVING CANONICAL BUSINESS OBJECT OPERATION NET

It is noted that we only discuss how to derive the canonical operation net of a certain object in a business process. With the similar method, the corresponding canonical object operation net for any other objects handled in a business process is also derived.

4.1 Task Operation Net

In order to characterize the operation information on a certain object in a process model, the particular task including operations on the object can be annotated with object's operation information. Here, we propose the task business object operation net (TBO2-net) to characterize the object operation information in task. Briefly, we call it task operation net.

Definition 4.1 (TBO2-net). A TBO2-net is an 8-tuple $< \Sigma_s, S_s, T_s, F_s, C_s, G_s, E_s, I_s >$, which is a subclass of WFCP-net. The concepts of all elements are similar with WFCP-net. In addition, specially pointing out:

The operation on the object is regard as a particular task type denoted as t^{op}. For $t \in T_s$, if the type of t is t^{op}, the corresponding guard function of t represents the precondition of operation t enabled. The expression function on the output arc of t denotes the post state of the object when operation t is performed.

For instance, the TBO2-net of task *designing* and *releasing* of the example in Section 2 are expressed in Figure 3 (b) respectively.

4.2 Business Object Operation Net

Now, we adopt refinement method (Suzuki & Murata, 1983; Betous-Almeida & Kanoun, 2004) to derive the business object operation net (BO2-net).

Definition 4.2 (Refinement). Let WFCP-net $N =< \Sigma, S, T, F, C, G, E, I >$, TBO2-net $N_s =< \Sigma_s, S_s, T_s, F_s, C_s, G_s, E_s, I_s >$, $T^R \subseteq T$, T^R is a task set of refinable task type in N. $\forall t \in T^R$, r_i denotes the input

place of task t, r_o denotes the output place of task t, satisfying: $\cdot r_o = \{t\} = r_i \cdot$, $|\cdot t| = |t \cdot| = 1$. $\exists$ corresponding TBO2-net N_s, i is the source place of N_s , o is the sink place of N_s , $C_s \subseteq C$. A new refinement net N_r is acquired by using N_s to substitute t , $N_r =< \Sigma_r, S_r, T_r, F_r, C_r, G_r, E_r, I_r >$. N_r is called BO2-net , where:

(i) $\Sigma_r = \Sigma \cup \Sigma_s$.

(ii) Given p_i and p_o, which are fusion places to glue WFCP-net N and TBO2-net N_s together. p_i denotes input fusion place, in addition, p_o denotes output fusion place. Satisfying:
$\cdot p_i =\cdot r_i, p_{i\cdot} = i\cdot, C(p_i) = C(r_i) = C(i)$, and
$\cdot p_o =\cdot o, p_{o\cdot} = r_{o\cdot}, C(p_o) = C(r_o) = C(o)$.

(iii) $S_r = S \cup S_s \cup \{p_i, p_o\} - \{r_i, r_o, i, o\}$.

(iv) $T_r = T \cup T_s - \{t\}$.

(v) $F_r = F \cup F_s \cup (\{(p_i, x) | x \in i\cdot\} \cup \{(x, p_o) | x \in \cdot o\} \cup \{(x, p_i) | x \in \cdot r_i\} \cup \{(p_o, x) | x \in r_o\cdot\} - \{(r_i, t), (t, r_o)\} \cup \{(x, r_i) | x \in \cdot r_i\} \cup \{(r_o, x) | x \epsilon r_o\cdot\} \cup \{(i, x) | x \in i\cdot\} \cup \{(x, o) | x \in \cdot o)\}$.

(vi) $G_r = G \cup G_s$.

(vii) $E_r = E \cup E_s$.

Figure 3 shows the refinement process. The BO2-net from the process model of the example is illustrated in Figure 3(c).

4.3 Canonical Business Object Operation Net

In this section we introduce reduction rules for generating the canonical business object operation net from the business object operation net. The basic idea is removing the *redundant tasks* from the business object operation net and the canonical net contains pure operation information on the object.

Definition 4.3 (Redundant task). All tasks in a BO2-net are *redundant tasks* except the following task types: t^{op}, t^{start} and t^{end}. *Redundant task* is labelled by black rectangle.

The reduction rules are presented in Figure 4. The first two rules (a) and (b) show that a redundant task (transition) connecting two places may be removed by merging the two places, provided that tokens in the first place can only move to the second place. The rule (c) shows that a short loop of a redundant task may be removed. The rule (d) shows that two parallel redundant tasks can be merged into one. The rule (e) and (f) show that two redundant tasks can be combined into one provided that the place between them has no other links. The last two rules (g) and (h) are similar to rule (e) and (f) except the two tasks are both redundant tasks.

Figure 5 shows the reduction process for the example in Section 2.

The generated net describes the consecutive operation evolvement of a particular object in the business process. With existing model checking tools, e.g. SPIN (Holzmann G.J., 2003), we can implement the validation strategy. Due to space limitations, we do not intend to elaborate the concrete implement process.

5 RELATED WORK

The research area related with our work is constructing state consistency relations between a business process and the related data. The concept of object life cycle comes from object-oriented technology which concerns about *object life cycle inheritance* and requires consistent specialization of behavior (Kappel & Schrefl, 1991; Schrefl & Stumptner, 2002). Later, some researches focus on establishing a link between the business process models and the object life cycles in business process management. An approach for generating a process specification is introduced in (Küster et al., 2007). In this literature, a technique is proposed for generating a business process description from a given set of objects and their reference object life cycle. In (Ryndina et al., 2006), an approach is presented to establish the required consistency, and then, two consistency notions are defined to verify the consistency between the generated object life cycle and the reference object life cycle.

Our work presented in this paper is enlightened by (Ryndina et al., 2006). The difference of our work is that the reference object life cycle is isolated with the process model, therefore, the operations on an object and the tasks in a process is not one-to-one corresponding relationship. On the contrary, Ryndina et al. only considers one-to-one corresponding relation between the tasks in a process and the operations on an object. Thus, we propose a new approach to characterize the operation information of an object in a process model. And then, a series of algorithms based on Petri-nets are presented to derive the canonical operation net of the business object from the business processes.

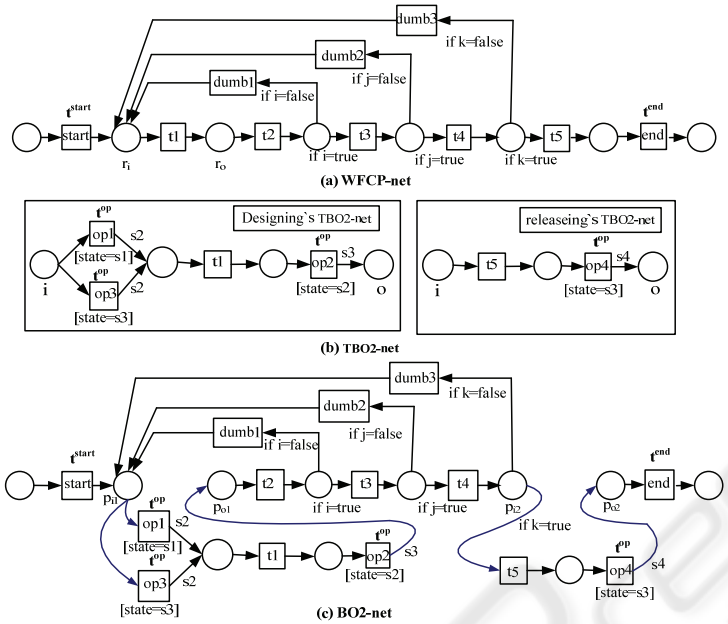

Figure 3: Refinement of a WFCP-net.

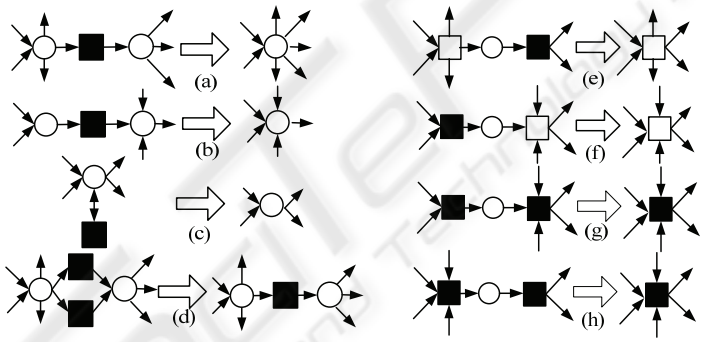

Figure 4: Reduction rules for redundant transitions.

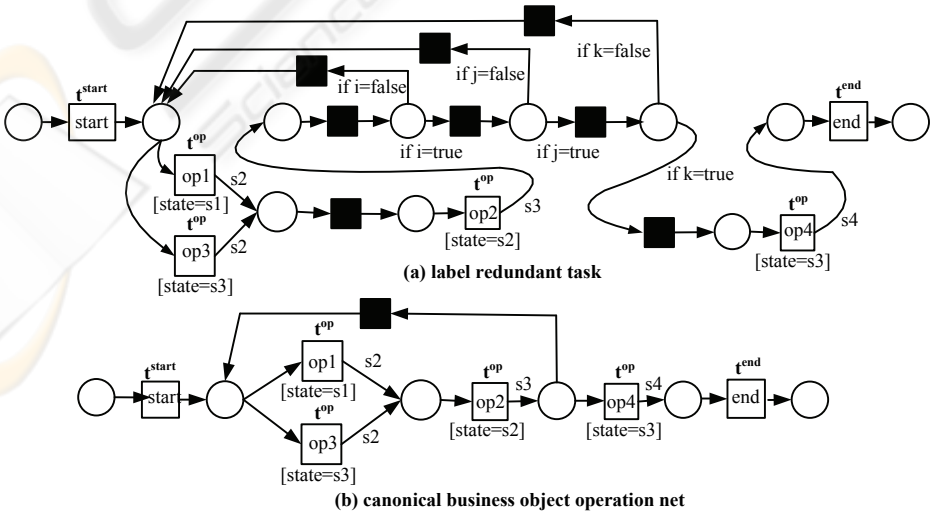

Figure 5: Deriving canonical business object operation net (Redundant tasks are shown as black rectangles).

6 CONCLUSIONS

In this paper, we have proposed an approach to derive the canonical object operation net. Firstly, we have described the object operations of a certain task by task operation net. Then, the business object operation net has been generated by replacing the task in process model based on WFCP-net with the corresponding task operation net. In the end, we have represented the reduction rules to derive the canonical object operation net. The approach can support the object operation information in the process model and derive the canonical object operation net from the process model even if the operations on the object is not one-to-one corresponding with the tasks in the process model.

As the future work, we intend: (1) to validate the consistency between the canonical object operation net from a process model and the reference object life cycle. (2) to extend the approach to support operation information description and consistency analysis of multi-objects processing in a process and multi-objects in collaboration processes. (3) to develop a prototype system to implement the validation strategy.

ACKNOWLEDGEMENTS

We are grateful to Tsinghua InfoTech Co., Ltd for their business process models accumulated in TiPLM system. This work is supported by the 973 National Basic Research Program of China (No. 2009CB320700), the 863 High-Tech Development Program of China (No. 2007AA040607), the 863 High-Tech Development Program of China (No. 2008AA042301) and the Program for New Century Excellent Talents in University.

REFERENCE

Aalst, W. M. P. V. D., 1998. The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1), 21-66.

Betous-Almeida, C. & Kanoun, K., 2004. Construction and stepwise refinement of dependability models. *Performance Evaluation.* 56, 1-4 (Mar. 2004), 277-306.

Clarke, E., Grumberg, O., and Long, D., 1996. Model checking. In *Proceedings of the NATO Advanced Study institute on Deductive Program Design (Marktoberdorf, Germany, July 26 - August 07, 1994). M. Broy, Ed.* Springer-Verlag, New York, Secaucus, NJ, 305-349.

Holzmann G.J., 2003. *The SPIN model checker: primer and reference manual.* Addison-Wesley, 2003.

Jensen K., 1997. *Colored petri nets : basic concepts, analysis methods and practical use. Basic Concepts*, Vol. 1, Berlin: Springer Verlag, 1997.

Kappel, G. & Schrefl, M, 1991. Object/behavior diagrams. In *Proceedings of the Seventh International Conference on Data Engineering* (April. 1991). IEEE Computer Society, Washington, DC, 530-539.

Küster, J.M., Ryndina,K., Gall,H., 2007. Generation of business process models for object life cycle compliance,In: *Proceeding of BPM2007*, Lecture Notes in Computer Science, vol. 4714, Springer, Berlin, 2007, 165-181.

Liu, D., Wang, J., Chan, S. C., Sun, J., and Zhang, L., 2002. Modeling workflow processes with colored Petri nets. *Computers in Industry*, 49(3)(Dec. 2002), 267-281.

Lee, G., Eastman, C. M., and Sacks, R., 2007. Eliciting information for product modeling using process modeling. *Data & Knowledge Engineering,* 62(2)(Aug. 2007), 292-307.

Ryndina, K., Küster, J. M., and Gall, H., 2006. Consistency of business process models and object life cycles. In: *Proceedings of the 1st Workshop on Quality in Modeling co-located with MoDELS 2006*, Technical report 0627, Technische Universiteit Eindhoven, 2006.

Suzuki, I. & Murata, T., 1983. A method for stepwise refinement and abstraction of Petri nets. *Journal of Computer and System Sciences,* 27(1), 1983, 51-76.

Stumpiner, M., Schrefl, M., 2000. Behavior Consistent inheritance in UML. In: *Conceptual Modeling – ER 2000,* LNCSLecture Notes in Computer Science, vol. 1920, Springer, Heidelberg 2000, 527-542.

Schrefl, M. & Stumptner, M., 2002. Behavior-consistent specialization of object life cycles. *ACM Transitions on Software Engineering and Methodology,* 11(1) (Jan. 2002), 92-148.

Sun, S. X., Zhao, J. L., Nunamaker, J. F., and Sheng, O. R., 2006. Formulating the data-flow perspective for business process management. *Information Systems Research,* 17(4) (Dec. 2006), 374-391.

TiPLM. (2008) *www.thit.com.cn/TiPLM/TiPLM.htm.*