Rigorous Business Process Modeling with OCL

Tsukasa Takemura[1,2] and Tetsuo Tamai[2]

[1] Software Development Laboratory, IBM Japan, Ltd.,
Yamato-shi, Kanagawa-ken, Japan,
tsukasa@jp.ibm.com
[2] Graduate School of Arts and Sciences, The University of Tokyo,
Tokyo, Japan,
tamai@graco.c.u-tokyo.ac.jp

Abstract. Recently business process modeling is getting a lot of attention as a predominant technology to bridge Business-IT gaps. UML activity diagram was drastically changed in UML 2.0 to support business process modeling. However, this emerging technology is insufficient to bridge the business-IT gaps as we expected because;

- Business process modeling is not so simple as business persons expected.
- Business process models don't provide enough information about activities' behavior.

In this paper, we show that;

- Rigorous description of business helps derivation of business process models.
- Rigorous business process models help to bridge the business-IT gaps.

We propose to use OCL to describe business and to model business process rigorously. We also show necessity of extension of OCL to express constraints of business process models and propose an extension of OCL for business process modeling.

1 Introduction

Enterprises must change their business processes in response to changes in their business enthronement to survive in the intensely competitive society in these days. When enterprises change their business processes, their IT systems also need to be changed according to the changes in their business processes.

Description about the business process provided by business persons tends to be ambigious and does not provide IT persons with enough information to build/change IT systems. This issue is called "business-IT gap", and this gap makes it difficult to change IT systems rapidly according to changes in the business. It is expected that a new technology to bridge business and IT becomes available.

Recently business process modeling is getting a lot of attention as a predominant technology to bridge Business-IT gaps. Business process modeling is

a method to model activities and their sequence, business objects passed among activities, resources, time, and cost required to perform the activities[3].

This emerging technology is not powerful enough to bridge the business-IT gaps as we expected because business process models provided by business persons are tend to be ambiguous. Many business consultants use presentation tools or drawing tools to depict business processes, and the business processes described in this manner have no formal syntax nor semantics. It is one of the reasons to make business process models ambiguous. This ambiguity makes it difficult for IT persons to understand the business process models and realize them as IT systems.

To bridge the business-IT gaps, business process models need to be more rigorous. To make business process models rigorous, we need to define formal syntax and semantics for business process models. One reason of drastic changes in UML activity diagram in UML 2.0[5] was to support business process modeling. Its foundation was changed from state chart diagram to Petri-net. This change allows us to describe data flows and parallel execution of activities in UML 2.0 activity diagrams.

Despite the change, business process models expressed in UML 2.0 activity diagram is still insufficient to bridge the business-IT gap because behavior of each activity or action is not expressed in the activity diagrams. Activities and actions in a business process model need to be implemented as application programs or Web services and so on in order to realize the business process on IT systems. To implement the activities or the actions, their behaviors need to be expressed in the business process model. Without this information, IT persons have no mean to realize the activities nor the actions on IT systems. OCL is thought as the best way to express such information in UML activity diagrams, however, the current OCL is not ready to be used in UML activity diagrams and we propose to be extended to add such information to business process models.

Present methodologies and notations of business modeling assume that it is not complicated work to model AsIs business processes by using control flows and data flows such as UML activity diagram. In some cases, business processes consist of asynchronous activities repeated with different intervals. It is difficult to model such processes only by connecting activities with control flows and object flows. We need to establish a modeling methodology for such complicated asynchronously repeated processes.

2 Case study

2.1 The company

The company in this case study is in manufacturing industry. It obtains materials from suppliers, builds up products from materials, and delivers products to customers.

When the company receives a customer order from a customer, it returns a promise for the customer order to the customer in order to ensure delivery of

products on the expected delivery date specified in the order. A customer order from customers includes product type, quantity, and expected delivery date. A promise also includes includes product type, quantity, and estimated delivery date. For correlation, a promise has the same ID as its associated customer order. In this paper, this business process is called "Order Promising".

The product is made of some materials obtained from suppliers, and the company issues purchase orders to suppliers and receives promises for them from suppliers to obtain the materials. Fig.1 shows Business Use Case diagram for this company's business process, "Order Promising".

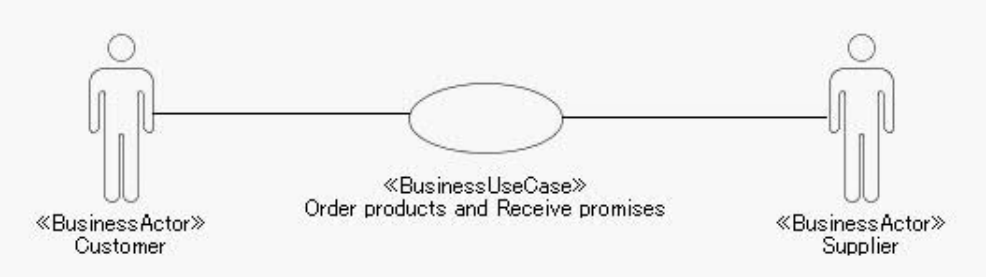

Fig. 1. Business Use Case for Order Promising

This company receives two types of orders; one is "customer order", and another is "forecast". A customer order is a firm order and customers can not cancel it while a forecast shows customer's intention to order products in a specific time frame. The company returns "promise for customer order" when it receives a customer order but does not return promise for "forecast".

The company creates the following three types of plans to prepare materials and satisfy customer orders, and issues purchase orders for materials to satisfy plans. Each plan this company creates includes product type, quantity, and execution complete date.

Resource Plan (RP) Long term plan including productive facilities, allocation of human resources, etc.

Production Plan (PP) Intermediate term plan including provision of long lead time materials

Master Production Schedule (MPS) Short term plan including provision of short lead time materials

The product is made of two types of materials; One is long lead time (LLT) material and another is short lead time (SLT) material. The purchase orders are issued to satisfy the plans.

Fig.2 shows Business Entities in this company and passed between the company and external entities.

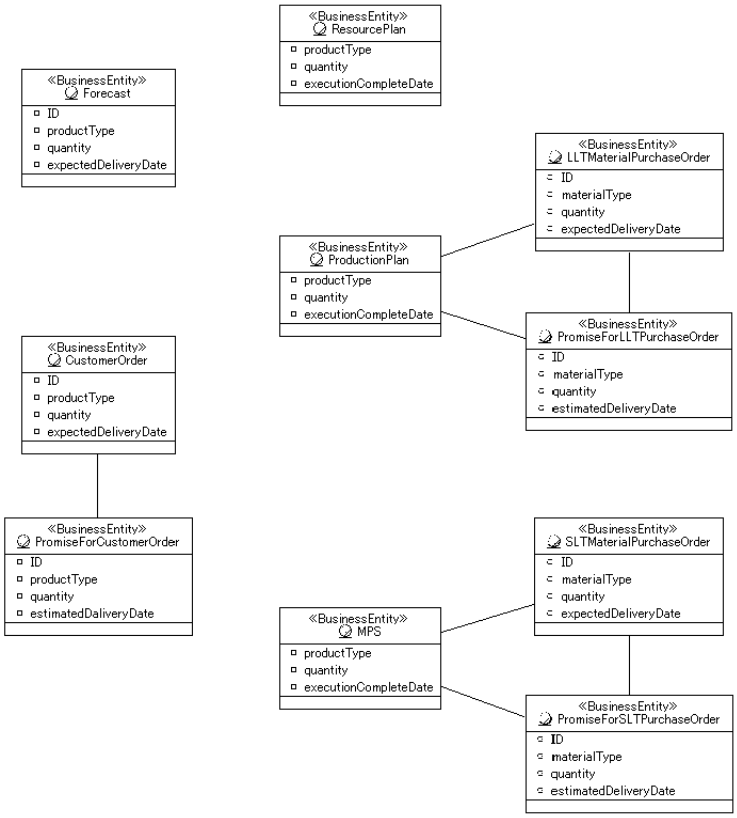

Fig. 2. Business Entities in Order Promising

This business process have predetermined time duration so-called order lead time and planning horizon. Planning horizon is duration from start of execution of a plan to completion of it. Order lead time is duration from issue of a order to expected time of products/materials delivery. Each type of order and plan has a specific lead time or a horizon.

In this case study, we use the following lead times, and planning horizons;

`FORCASTLEADTIME` Forecast lead time, the shortest duration from receiving a forecast to product delivery associated to the forecast.

`CUSTOMERLEADTIME` Customer lead time, the shortest duration from receiving a customer order to product delivery associated to the order.

`LLTMATERIALLEADTIME` Lead time for the long lead time material, the shortest duration from issueing a purchase order for the long lead time material to receiving the material.

`SLTMATERIALLEADTIME` Lead time for the short lead time material, the shortest duration from issueing a purchase order for the short lead time material to receiving the material.

`MPSHORIZON` MPS horizon, the duration from start of execution of a MPS to completion of execution of the MPS.

`PPHORIZON` Production Plan horizon, the duration from start of execution of a production plan to completion of execution of the production plan.

`RPSHORIZON` Resource Plan horizon, the duration from start of execution of a resource plan to completion of execution of the resource plan.

2.2 Order Promising

In this business process, a promise for a customer order is returned in order to ensure delivery of products on the expected delivery date specified in the customer order. To ensure it, a concept, available to promise (ATP) is used in this business domain. ATP is the amount of products available on a specific date including future. It can be calculated by subtracting accumulated amount of promised delivery by the specific date from accumulated amount of products planned to be produced by the specific date(Fig.3). Fig.4 shows ATP conceptually. As is clear from Fig.4, ATP varies on magnitude relation of lead times and planning horizons. From the domain knowledge depict in this figure, it is unveiled that ATP consists of what already provisioned by executed and executing plans and what already committed to promises, and ranges of plans and promises included in ATP are depend on the expected time of delivery.

To ensure delivery of requested products on the expected delivery date, the quantity the company can promise is less than ATP. "Order Promising" is thought as a process to allocate products to a customer order from ATP. In other words, ATP is the least upper bound of a promise for a customer order.

ATP is a key concept of "Order Promising" and it is required for "Order Promising" business process model to express what APT is to transfer the domain knowledge about this process to IT persons.

3 Business modeling

3.1 Business object model

The domain knowledge described rigorously brings out what are inputs to activities and what are outputs from activities. These inputs and outputs can be modeled in a business object model (a kind of class diagram with an UML profile for business process modeling) as shown in Fig.5. In this business object model, classes with ≪BusinessEntity≫ stereotype represent business entities, A class with ≪business process≫ stereotype is not a business entity but an artificial class introduced to specify a context for OCL expressions. By introducing this context in business entity model, the domain knowledge described in the previous section can be expressed in OCL expressions as follows.

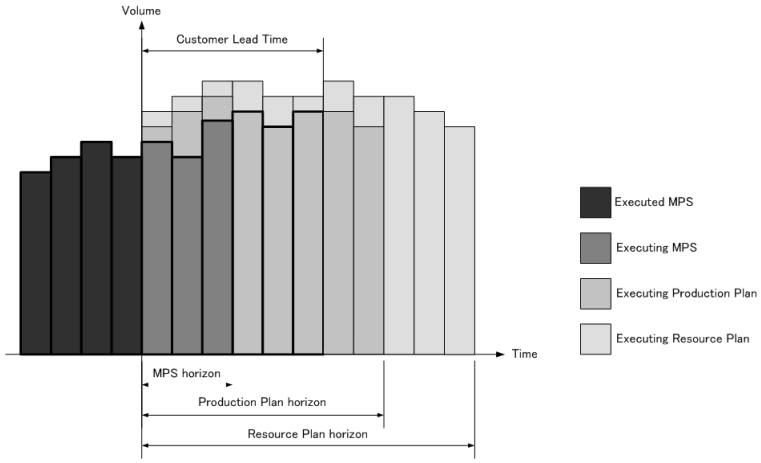

Fig. 3. Rectangles surrounded by bold lines indicate accumulated amount of products planned to be produced within Customer Lead Time

Note that capitalized words in these OCL expressions represent process scoped constant values or operations.

```
context AvailableToPromise::getQuantity( date : Date ) : Integer
body:
existingPlans.getQuantity( date )
- existingPromises.getQuantity( date )

context ExistingPlan::getQuantity( date : Date ) : Integer
body:
mpss->select( executionCompleteDate <= date ).quantity->sum()
+ productionPlans->select( executionCompleteDate <= date and
executionCompleteDate - TODAY() > MPSHORIZON ).quantity->sum()
+ resourcePlans->select( executionCompleteDate <= date and
executionCompleteDate - TODAY() > PPHORIZON ).quantity->sum()

context ExistingPromise::getQuantity( date : Dime ) : Integer
body:
promisesForCustomerOrders->select( estimatedDeliveryDate <= date and
estimatedDeliveryDate - TODAY() > CUSTOMERLEADTIME ).quantity->sum()

context OrderPromising::promiseOrder( )
pre:
newCustomerOrder.expectedDeliveryDate >= TODAY() + CUSTOMERLREADTIME and
// Lead time of the new customer order is longer than CUSTOMERLREADTIME
newPromisesForCustomerOrder->select( ID = newCustomerOrder.ID )->isEmpty()
```

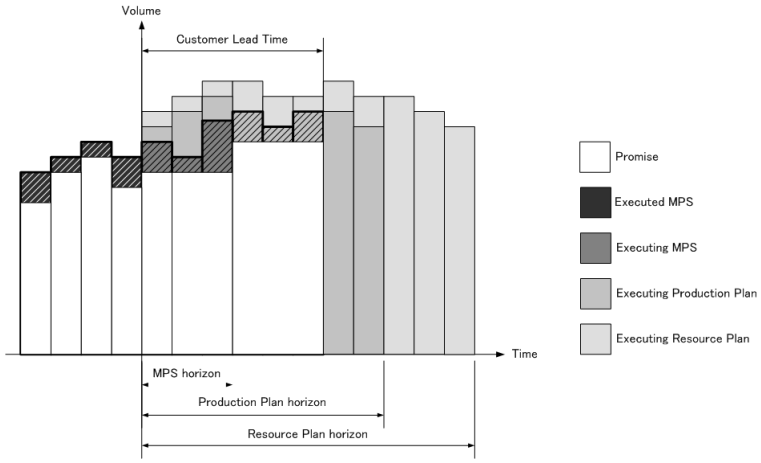

Fig. 4. Rectangles with hatching indicate Available to promise within Customer Lead Time

```
// no promise for the customer order exists
post:
newPromisesForCustomerOrder->select( ID = newCustomerOrder.ID )->notEmpty()
// if there are any promise for the customer order
implies
newPromisesForCustomerOrder->select( ID = newCustomerOrder.ID )->size() = 1 and
// there is only one promise for the customer order and
newPromisesForCustomerOrder->select( ID = newCustomerOrder.ID ).quantity->sum()
= newCustomerOrder.quantity and
// total quantity of promises is equal to requested quantity in the customer order and
newPromisesForCustomerOrder->select( ID = newCustomerOrder.ID )->
forAll( estimatedDeliveryDate = newCustomerOrder.expectedDeliveryDate ) and
// estimated delivery date of the promise is equal to
// expected delivery date of the customer order and
newPromisesForCustomerOrder->select( ID = newCustomerOrder.ID )->
forAll( quantity <= availableToPromise@pre.getQuantity( estimatedDeliveryDate ) )
// total quantity of promises is equal to or less than ATP
```

3.2 Business process model with constraints

When the business process is modeled in UML activity diagram, "promise order" is modeled as an activity or an action. In this paper, we assume that "promise order" is modeled as an action. From the description in the previous section, we modeled "promise order" as an action with five input pins and one output pin as shown in Fig.6.

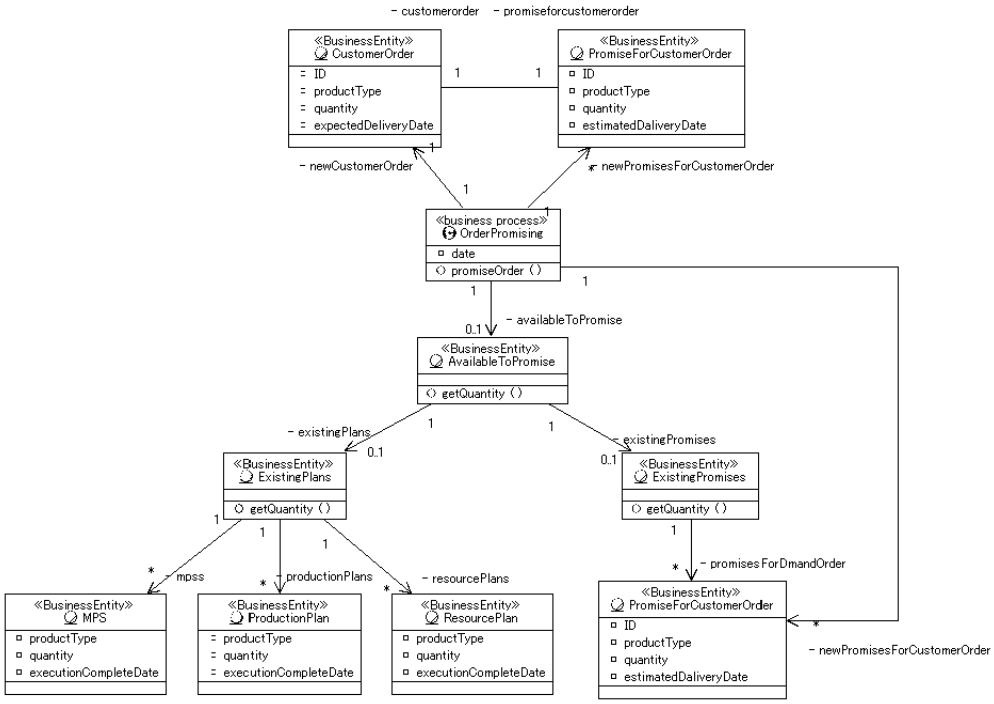

Fig. 5. Business object model for Order Promise

Most commonly, activity diagrams are created prior to a business object model in business process modeling because activity diagrams are more intuitive for business persons like business consultants and their customers.

When an OCL expression is attached to the action, it is required to express the equivalent information as one expressed by OCL attached to the business object model in Fig.5. Especially, we need to describe constraints on output pins by using values of input pins. It means that the OCL expression specifies the action as a context and refers to the pins like as attributes of a class. It requires to extend OCL as we explain in the next section.

3.3 Customization

In this section, we show that the business process can be customized by using constraints defined in the business environment.

Let's say the company in this case study has defined the planning horizons and the lead times as follows;

```
MPSHORIZON < CUSTOMERLAEDTIME < PPHORIZON < FORECASTLEADTIME < RPHORIZON
```

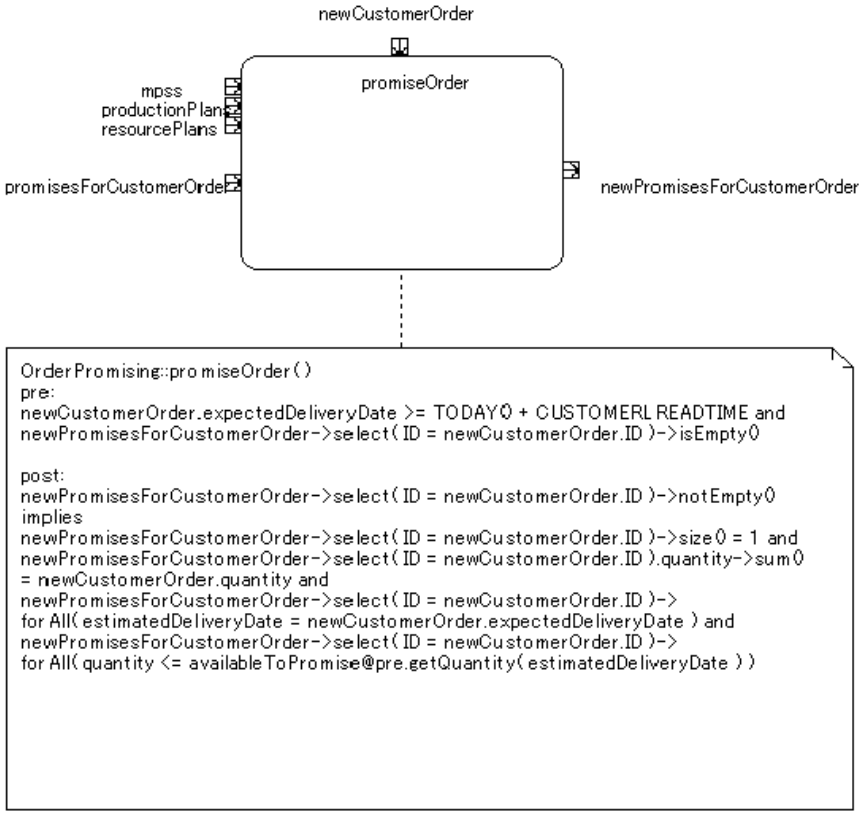

Fig. 6. PromiseOrder action with constraints in activity diagram

By using the magnitude relation `PPHORIZON > CUSTOMERLEADTIME` and an equation `newCustomerOrder.expectedDeliveryDate = TODAY() + CUSTOMERLEADTIME` in the postcondition of promiseOrder, the last term in body of `ExistingPlans.getQuantity` can be reduced as follows;

```
 resourcePlans->select( executionCompleteDate <= date and executionCompleteDate - TODAY()
> PPHORIZON ).quantity->sum()
  ⇓
 resourcePlans->select( executionCompleteDate <= date and executionCompleteDate - TODAY()
> PPHORIZON ).quantity->sum()
  ⇓
 resourcePlans->select( executionCompleteDate <= date and executionCompleteDate > TODAY()
+ PPHORIZON ).quantity->sum()
```

```
  ⇓
 resourcePlans->select( executionCompleteDate <= date and executionCompleteDate > TODAY()
+ CUSTOMERLEADTIME ).quantity->sum()
  ⇓
 resourcePlans->select( executionCompleteDate <= date and executionCompleteDate >
newCustomerOrder.expectedDeliveryDate ).quantity->sum()
  ⇓
 resourcePlans->select( executionCompleteDate <= date and executionCompleteDate > date
).quantity->sum()
  ⇓
 resourcePlans->select( false ).quantity->sum()
  ⇓
 0
```

This reduction indicates that ResourcePlan does not contribute to Available-ToPromise for this action in this business environment, and we can eliminate input pins from promise order action. As in a case of promise order action, some inputs pins and output pins can be eliminated from some other actions. This suggests that constraints for actions help customization of a business process in a specific business environments.

3.4 Composition

Actions modeled in this method have rigorously defined input pins and output pins, these actions can be composed into one business process by connecting pins to common data stores. Fig.7 shows the composed business process model. The gray object flows in this figure denote the object flows connected to the pins eliminated on the customization.

As a result of this composition, a complicated business process where actions are repeated asynchronously can be modeled in an UML activity diagram.

4 OCL extension for business process modeling

As shown in the previous section, OCL is useful for rigorous business process modeling. However, OCL is required to be extended to add the constraints described with OCL to UML activity diagrams. In this section, we explain the required extension for business process modeling using OCL and UML activity diagrams.

4.1 Contexts

To describe behavior of actions, it is required to express preconditions and post-conditions of actions. Consequently, OCL expression needs to have a context from which we can navigate to input pins and output pins of actions in order

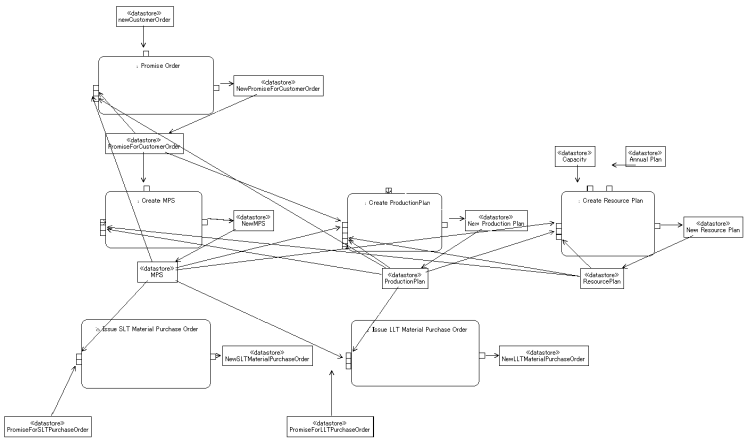

Fig. 7. Composed Business Process Model

to express the constraint of the pins. The action is the best candidate of the context of OCL expression because the action owns the pins.

UML metamodel[5] allows to add Constraint to Actions, however, OCL specification[4] does not expect to specify an action as a context.

The chapter 7.3.4 of OCL specification denotes;

The OCL expression can be part of Precondition or Postcondition, corresponding to ≪precondition≫ and ≪postcondition≫ stereotypes of Constraint associated with an Operation or other behavioral feature. The contextual instance self then is an instance of the type which owns the operation or method as a feature.

First of all, neither Action nor Activity is BehavioralFeature. Even if Constrains are allowed to associate to Actions and Activities, an owner of Actions and Activities is an entity which is executing the business process[8]. The entity executing the business process is a company or an enterprise, and they are not appropriate as a context of OCL expression because we don't model such entities in business process models.

We propose to extend OCL to allow to specify an Action as a context of OCL expression and navigate to its input/output pins.

4.2 @pre

To describe behavior of actions, it is required to express constraints on outputs of actions. Consequently, constraints on objects on output pins need to be expressed by value of objects on input pins. It also required to express the constraints by using the value of inputs on their arrival on the input pins. To refer the value of an input object on arrival on an input pins, it is natural to use "@pre" keyword.

We propose to extend OCL to allow to use the keyword for this purpose.

4.3 Example

By these extensions, we can express constraints on objects on output pins by using value of objects on input pins in the similar way as constraints on operations.

```
context promiseOrder
pre:
newCustomerOrderPin.expectedDeliveryDate >= TODAY() + CUSTOMERLRADTIME and
newPromisesForCustomerOrderPin->select( id = newCustomerOrderPin.ID )->isEmpty()
post:
newPromisesForCustomerOrderPin->select( id = newCustomerOrderPin.ID )->notEmpty()
implies
newPromisesForCustomerOrderPin->select( id = newCustomerOrderPin.ID@pre )->size() =
1 and
newPromisesForCustomerOrderPin->select( id = newCustomerOrderPin.ID@pre ).quantity->sum()
= newCustomerOrderPin.quantity@pre and
newPromisesForCustomerOrderPin->select( id = newCustomerOrderPin.ID@pre )->
forAll( estimatedDeliveryDate = newCustomerOrderPin.expectedDeliveryDate@pre ) and
newPromisesForCustomerOrderPin->select( id = newCustomerOrderPin.ID@pre )->
forAll( quantity <= availableToPromise.quantity@pre( estimatedDeliveryDate ) )
```

5 Related works

As a notation for business process, UML and its extensions are proposed[1][2][7]. Most notations use UML activity diagrams or similar diagram to express activities and their sequence. Modeling methods for these notations assume that persons related to business processes understand the flow of their business process. Our approach does not assume this situation because object flows and control flows of some business process are very complicated and difficult to understand. In our approach, complicated business processes as shown in this paper can be modeled rigorously.

Koubarakis and Plexousakis[6] proposed to model an organization and its business process formally, and showed importance of enterprise model.

6 Conclusion

In this paper, we showed that rigorous description about the business and its related domain knowledge helps creation of business process models. And the created business process models are also rigorously described because preconditions and postconditions of each action are rigorously described by OCL. IT persons can use the preconditions and postconditions to implement the action in the business process on IT systems because the model describes what the action should perform and should not perform formally. If the business process is implemented on Service Oriented Architecture (SOA), the preconditions and postconditions depict requirements for a service.

By describing business environments rigorously, the business process model can be customized systematically. It helps rapid transition of a business process and its IT systems. We can expect the business process models help to bridge Business-IT gaps.

We proposed extension of OCL to support our approach. To describe constraints on actions in UML activity diagram, it is appropriate to specify an action as a context of OCL expression. To support this notation, we proposed extension of OCL.

As shown in this paper, OCL is a powerful tool for business process modeling. But, its syntax and semantics are still difficult to learn for business persons. To resolve this issue,dijkman2002algorithm we are planning to define domain specific languages to simplify OCL for business persons.

References

1. Nuno Castela, Jose M. Tribolet, Alberto Silva, and Arminda Guerra. Business process modeling with UML. In *3rd International Conference on Enterprise Information Systems*, Vol. 2, pp. 679–685, Setubal, Portugal, July 2001.
2. Hans-Erik Eriksson and Magnus Penker. *Business Modeling With UML Business Patterns at Work*. John Wiley & Sons Inc., 2000.
3. Jaap Gordijn, Hans Akkermans, and Hans van Vliet. Business modelling is not process modelling. In *Conceptual modeling for e-business and the web (ECOMO-2000)*, pp. 40–51, Salt Lake City, USA, October 2000. Springer-Verlag.
4. Object Management Group. Ocl 2.0 specification version 2.0. `http://www.omg.org/cgi-bin/doc?ptc/05-06-06`, 2005.
5. Object Management Group. Uml 2.0 superstructure specification. `http://www.omg.org/cgi-bin/doc?formal/05-07-04`, 2005.
6. Manolis Koubarakis and Dimitris Plexousakis. A formal model for business process modeling and design. In *Conference on Advanced Information Systems Engineering*, pp. 142–156, 2000.
7. Chris Marshall. *Enterprise Modeling with UML:Designing Successful Software Through Business Analysis*. The Addison-Wesley Object Technology Series. Addison-Wesley, 1999.
8. Jos B. Warmer and Anneke Kleppe. *The Object Constraint Language: Getting Your Models Ready for Mda*. The Addison-Wesley Object Technology Series. Addison-Wesley, second edition, 2003.