Workshop: Databases and Software Engineering

Title: Object-Oriented Conceptual Modeling for Collaboration Management in Virtual Enterprises

Authors: Jyhjong Lin and Tsui-e Lin

Department of Information Management

Chaoyang University of Technology

Wufeng, Taichung County, Taiwan

E-mail: {jjlin@mail.cyut.edu.tw, linlingo@mail.ht.net.tw}, Fax: 886-4-23742337

Contact author: Jyhjong Lin

Department of Information Management

Chaoyang University of Technology

Wufeng, Taichung County, Taiwan

E-mail: jjlin@mail.cyut.edu.tw, Fax: 886-4-23742337

Abstract

For the rapid progress of Internet technologies in recent years, Electronic Commerce (EC) has gained attention as a major theme for enterprises to keep their competitiveness. From the perspective of effective resources utilization, it becomes now an important goal for an enterprise to promote its performance and competitiveness through integrating itself and relevant suppliers and consumers as a virtual group to achieve the so-called Business-to-Business EC. In this paper, we propose an object-oriented modeling approach that addresses the management of collaboration on the Internet between enterprises. The approach divides those required mechanisms for collaboration management into three layers: commitment, role, and activity ones. With this architecture, two enterprises may collaborate via the establishment and maintenance of commitment, the collaboration and coordination between roles, and the interaction and coordination between activities. For specification, an object-oriented model is presented for each layer that describes the working details of that layer. To illustrate, these models are applied in a simplified supply chain management application among various enterprises.

Keywords: virtual enterprise, collaboration management, object-orientation, conceptual modeling

Object-Oriented Conceptual Modeling for Collaboration Management in Virtual Enterprises

Jyhjong Lin and Tsui-e Lin
Department of Information Management
Chaoyang University of Technology
Wufeng, Taichung County, Taiwan
E-mail: {jjlin@mail.cyut.edu.tw, linlingo@mail.ht.net.tw}, Fax: 886-4-23742337

Abstract

For the rapid progress of Internet technologies in recent years, Electronic Commerce (EC) has gained attention as a major theme for enterprises to keep their competitiveness. From the perspective of effective resources utilization, it becomes now an important goal for an enterprise to promote its performance and competitiveness through integrating itself and relevant suppliers and consumers as a virtual group to achieve the so-called Business-to- Business EC. In this paper, we propose an object-oriented modeling approach that addresses the management of collaboration on the Internet between enterprises. The approach divides those required mechanisms for collaboration management into three layers: commitment, role, and activity ones. With this architecture, two enterprises may collaborate via the establishment and maintenance of commitment, the collaboration and coordination between roles, and the interaction and coordination between activities. For specification, an object-oriented model is presented for each layer that describes the working details of that layer. To illustrate, these models are applied in a simplified supply chain management application among various enterprises.

Keywords: virtual enterprise, collaboration management, object-orientation, conceptual modeling

1 Introduction

Conceptual modeling is an important step in developing a computer-based application that collects adequate user requirements about the application domain (e.g., the structural, behavioral, and possibly desirable control/safety aspects of the application). It has been recognized that failure to identify the real application domain knowledge may result in late delivery, poor quality, and high maintenance costs. In general, conceptual modeling can be done by using function-, data-, or object-oriented approaches where the development of object-oriented ones is particularly motivated by the drawbacks and problems in the other two kinds: the significant features and benefits of object-oriented models would make application software easier to be understood, maintained, and reused.

For the rapid progress of Internet technologies in recent years, Electronic Commerce (EC) has gained attention as a major theme for enterprises to keep their competitiveness. From the perspective of effective resources utilization, it becomes now an important goal for an enterprise to promote its performance and competitiveness through integrating itself and relevant suppliers and consumers as a virtual group to achieve the so-called Business-to-Business EC. As a common recognition, a software system that realizes such a virtual enterprise (VE) environment needs to explicitly capture and manage the functional and contractual relationships between participant enterprises in VE. This kind of systems in particular have to support the establishment and maintenance of contracts between two enterprises, and the interaction/coordination of working processes between these two enterprises. To deal more efficiently with these complex aspects, it is not uncommon to think of the powerful object-oriented paradigm that possesses such features as encapsulation of an object's specifics and interacted/coordinated nature of its behaviors; these features make an object-oriented approach easier to be configured for full supports of these aspects. To account for this, we propose in this paper an object-oriented modeling approach that

provides extensive supports for the specification of collaboration management between two VE members.

For an open environment as on the Internet, any contractual collaboration between two VE members must comply with certain prescribed rules under which each member plays an appropriate role(s). That is, each member has to have its processes strictly follow those rules with regard to that role(s) it plays to collaborate with the other member. In the literature, many approaches have been proposed for specifying these requirements [2-7,9-11,13,14,17,18,20]. Among them, in our knowledge, the commitment-based approach in [9,10] provides a most natural way to deal with how VE members may comply with the requirements of the role(s) they play. Based on this idea, our approach divides those required mechanisms for collaboration management into three layers: commitment, role, and activity ones - a commitment is joined by a set of participant roles where each role offers under required rules various capabilities (i.e., functionalities) to be realized by a set of working activities (i.e., processes). With this architecture, two enterprises may collaborate via the establishment and maintenance of contractual commitments, the collaboration and coordination between those roles under the commitments, and the interaction and coordination between those activities that realize the roles. For specification, an object-oriented model is presented for each layer that describes the working details of that layer: (1) a commitment model that specifies the joined roles and associated rules of commitments; (2) a role model that presents the capabilities of and the rule-complied collaboration/coordination between roles; and (3) an activity model that describes the behaviors of and the rule-complied coordination between activities.

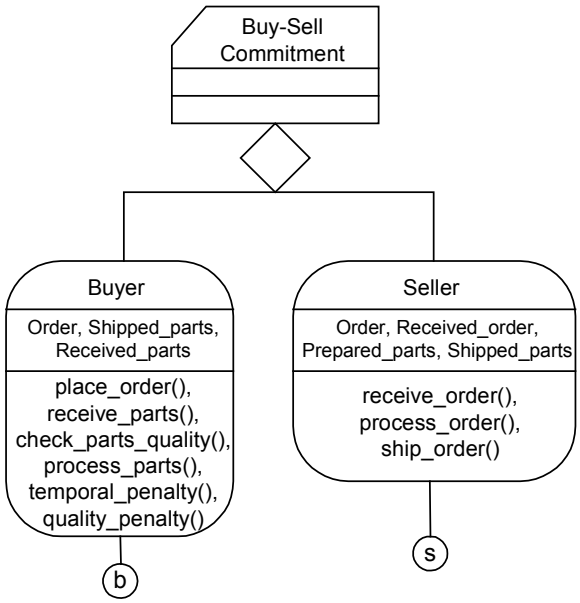
With these three models that describe the collaboration between VE members from a higher-level of abstraction to a lower-level of realization, the activity model in particular imposes formal constructs based on Petri nets [15,16] for verification of rule-compliance during the collaboration. In our knowledge, this is very critical for a VE environment, since no one is truly trustable in such an open environment. For illustration, the three models are applied to the specification of a simplified supply chain management application among various enterprises.

This paper is organized as follows. Section 2 overviews the background and motivation of the proposed approach. Section 3 presents the three models in the approach. A method that describes how these models are applied to the specification of a VE application will be presented in section 4. Finally, section 5 has the conclusions and future work.

2 Background and motivation

For an open environment as on the Internet, any contractual collaboration between two VE members must comply with certain prescribed rules under which each member plays an appropriate role(s). In the literature, there have already been many discussions related to this issue as in [2-7,9-11,13,14,17,18,20]. Among them, in our knowledge, the authors in [9,10] proposed a sophisticated SoCom (Sphere of Commitment) concept to address the management of commitments in VE. In SoCom, a set of roles and the commitments these roles must satisfy are defined. In particular, their representing and reasoning about commitments was prototyped using IBM's ABE [1] that includes a rule-based reasoning system with sets of declarative rules and facts. In general, SoCom provides a sound mechanism for collaboration management, but, by using a declarative approach, it lacks a visual formalism for specifying and verifying how the collaboration proceeds and satisfies desired commitments. As commonly recognized, however, a visual formalism for behavioral specification and verification provides a better conduit for comprehension and reasoning about the application being developed. Similar approaches can be found in [14,20] that used propositional temporal logic as its specification and verification tool. These approaches, anyway, support well only the specification of commitments; they do not provide sufficient mechanisms for collaboration management as presented in SoCom.

In contrast, the authors in [5,17] proposed an advanced approach for managing process and service



rule 1: @(b.place_order) --> $\Diamond_{[10,10]}$@(b.receive_parts) V $\Diamond_{[11,12]}$@(b.tempal_penalty)
rule 2: @(b.receive_parts) --> $\Diamond_{[3,4]}$@(b.process_parts) V $\Diamond_{[2,4]}$@(b.quality_penalty)

Figure 1: specification of a commitment with roles and rules

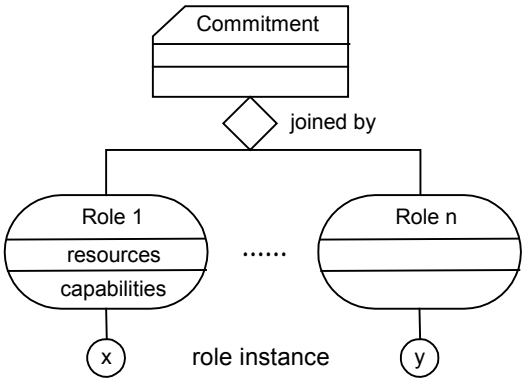


Figure 2: graphical symbols of the commitment model

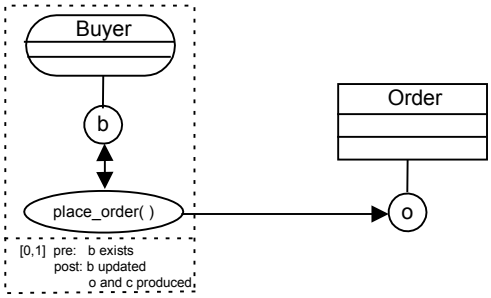


Figure 3: specification of a capability of a role

fusion in VE. Their CMI infrastructure provides a well visual way to specify the management of collaboration processes; a state transition diagram (STD) is adopted and extended for specifying the behaviors of participant VE members. Although this approach addresses very well and completely collaboration management in VE, it does not deal with the needs of specifying and verifying how the collaboration proceeds and satisfies desired commitments.

Our method is proposed to supplement the abovementioned deficiencies in these approaches by combining the specification and satisfaction of commitments with the modeling of collaborative behaviors in VE. It uses an object-based timed temporal logic OTTL [12] to specify desired rules, and employs an object-oriented model to specify under these rules the behaviors of participant VE members. In order to deal with the complexity of modeling contractual collaboration, the object-oriented model supports the specification in a top-down fashion. As results, a higher-level commitment model is created first that describes effectively the joined roles and associated rules of a commitment without considering detailed specification. That is, the detailed specification via role and activity models start after all roles and rules have been described in an abstract level. We think this provides better understanding about collaboration management before proceeding too early to formally specify them using some complex notations. Finally, due to its formal semantics of the activity model, behavioral verification of satisfying those desired rules in OTTL can be conducted via formal analysis of the model [12].

3 Modeling constructs

The modeling constructs of our approach include three models: (1) a commitment model that specifies the joined roles and prescribed rules of a commitment; (2) a role model that presents the capabilities of and the rule-complied interaction/coordination between roles; and (3) an activity model that describes the behaviors of and the rule-complied coordination between activities.

3.1 The commitment model

By the definition of a commitment using roles [9,10], the commitment model specifies the joined roles of a commitment. Two kinds of object types are used in the model: commitment ones and role ones. Figure 1 shows an example model that specifies by appropriate object types a commitment and its constituent two roles. Each role comes also with a set of capabilities it provides and the resources these capabilities may access. The small circle attached to each role (object type) refers to an instance of the role with which instances of other roles are able to interact. In addition to the structural aspect of the commitment, a set of rules associated with the commitment that these roles must comply with is also specified by using the statements in OTTL [12]. OTTL is an object-based timed temporal logic that is defined for specifying and verifying the temporal and safety constraints between objects. In our view, it is also well able to address the temporal and obligation constraints that are commonly desirable between two collaborative roles. For illustration, the rule 1 specifies a temporal constraint between a buyer and a seller: an order issued by the buyer needs to be processed and returned by the seller with desired parts at the 10th time unit (e.g., day) or a temporal penalty will be issued by the buyer within 11 to 12 days. The rule 2 is an obligation constraint: after receiving desired parts, the buyer will process the parts within 3 to 4 days or for some quality reasons, a quality penalty will be raised within 2 to 4 days. The reader is referred to [12] for more details about OTTL. Figure 2 has the graphical symbols of the commitment model.

3.2 The role model

With a commitment model, the role model is used to address more details about the capabilities of the roles in the commitment model. It describes how these capabilities access relevant resources to achieve desired functionality and how they interact/coordinate with each other under the rules specified in the commitment model. The modeling constructs of the role model include four kinds of object types: role ones, resource ones, control ones, and fault ones. In particular, the resource objects specify those things or entities in the application domain where the capabilities of each role may access to achieve their functionality. The control objects are used to give control flows between capabilities in order to satisfy the prescribed rules. Lastly, the fault objects are used to model the detection of any failures, i.e., violations of the rules, occurred during the interaction between capabilities.

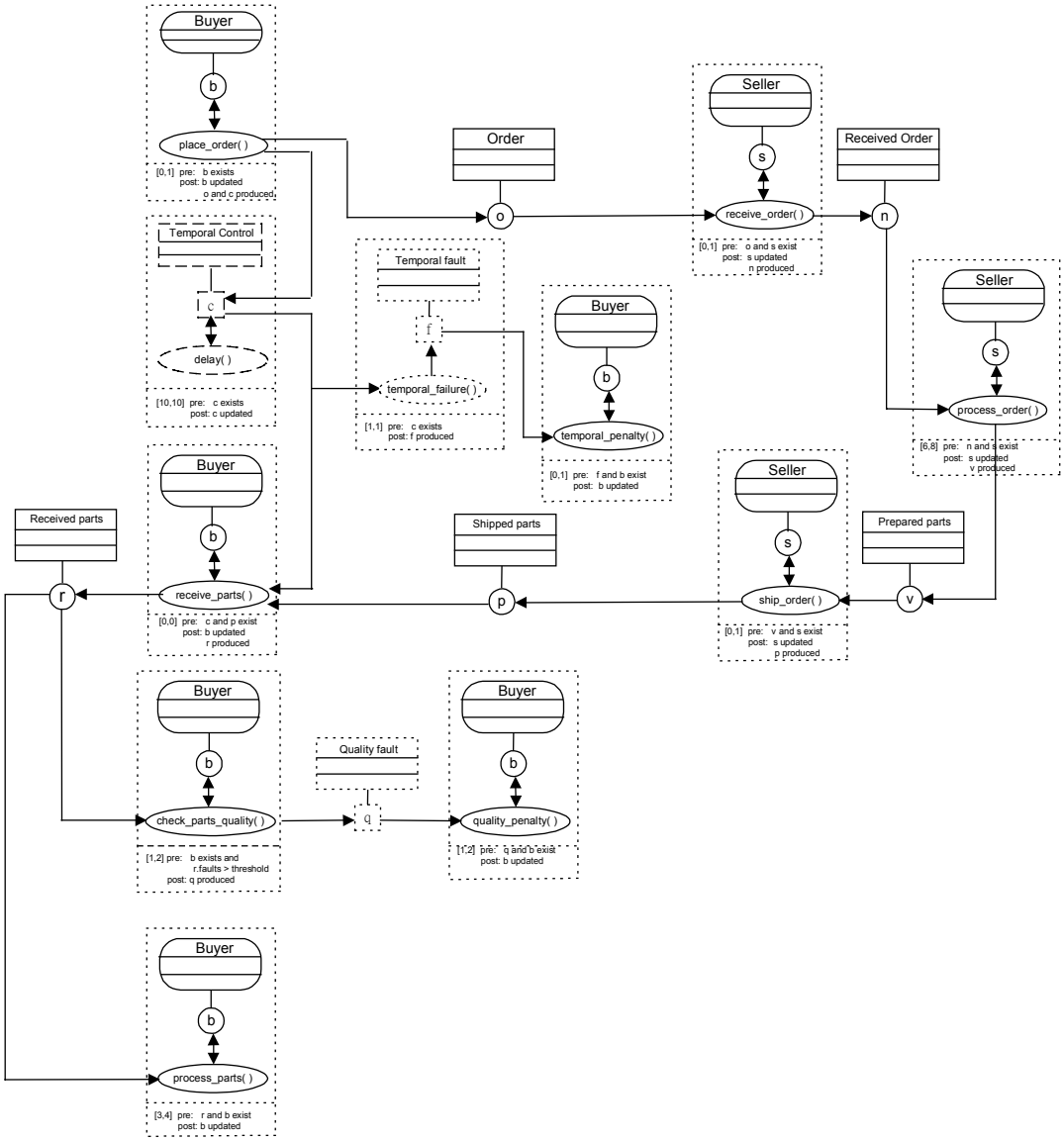


Figure 4: specification of roles and capabilities joined in a commitment

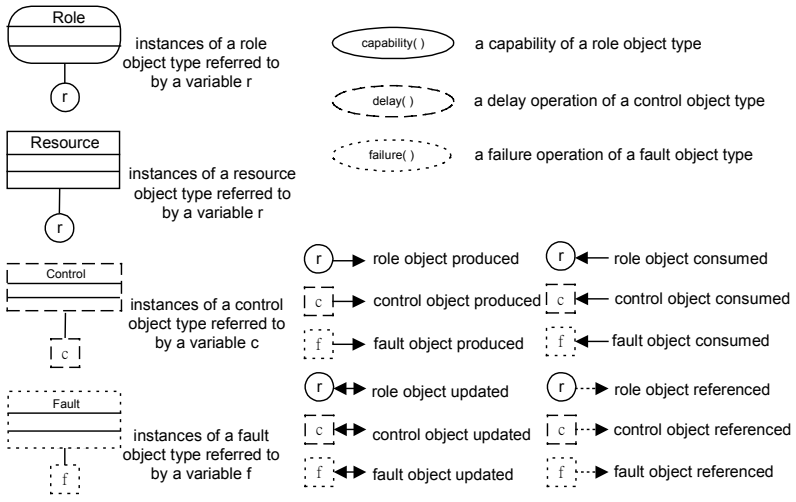


Figure 5: graphical symbols of the role model

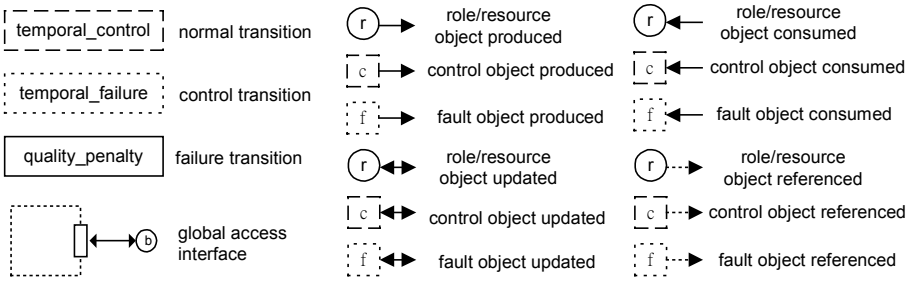


Figure 6: graphical symbols of the activity model

In the model, the capabilities of each role are specified one at a time, together with its execution effects on relevant resources. As shown in Figure 3, each capability is specified with a name, a set of interaction objects where its execution accesses, a pre/post-condition that its execution satisfies, and a time interval within which its execution takes place. With this specification, a capability is executable if and only if its pre-condition is true, and since then, its execution occurs within the specified time interval that makes its post-condition true. Figure 4 illustrates the specification of how the capabilities of a buyer and a seller behave and interact via relevant resources with each other to accomplish a simplified buy-sell cycle (the use of control and fault objects will be introduced below). Figure 5 shows the graphical symbols of the role model.

Control objects are used to give required control flows between capabilities in order to satisfy the collaborative rules defined in the commitment model. Their specification is similar to that for the roles; each control object type has operations to be specified one at a time for modeling of a time/temporal relationship between two capabilities. For illustration, in Figure 4 the delay( ) operation of the control object type Temporal_Control provides a modeling of 10 time units delay between the place_order( ) and receive_parts( ) capabilities of a buyer in order to satisfy the rule 1 described in Figure 1: an order issued by the buyer is expected to be returned with desired parts at the 10th day.

Fault objects are used to model the detection of any failures, i.e., violations of the rules, occurred during the interaction between capabilities. Their specification is also similar to that for the roles; each fault object type has operations to be specified one at a time for modeling of a failure occurred during the interaction between capabilities. For illustration, in Figure 4 the temporal_failure( ) operation of the fault object type Temporal_Fault provides modeling of the failure to the rule 1 described in Figure 1: an order issued by a buyer is expected to be returned with desired parts at the 10th day. In this case, a fault object f is produced that is to be used by the temporal_penalty( ) capability of the buyer for satisfying the second part of the rule 1: once the order cannot be returned at the 10th day, a temporal penalty will be issued within 11 to 12 days.

3.3 The activity model

With a role model, the activity model is used to address how the capabilities of the roles in the role model are realized by a set of working activities. In our approach, an activity is defined as a task that contributes to the fulfillment of a capability (or part of it). The modeling constructs of the activity model are based on Petri nets [15,16] with a set of transitions and places. Transitions specify the activities that occur in the system, which in turn have three kinds: normal transitions that describe the working processes occurred in the system, control transitions that impose the control flows between working processes, and failure transitions that capture the failures occurred during the interaction between working processes. Likewise, places are divided into three kinds: normal places that hold role or resource objects for the execution of transitions, control places that hold control objects for the control of the execution of transitions, and fault places that hold fault objects for the recognition of the failures occurred.

Each transition is specified with a name, a set of interaction places that its execution accesses, a pre/post-condition that its execution satisfies, and a time interval within which its execution takes place.

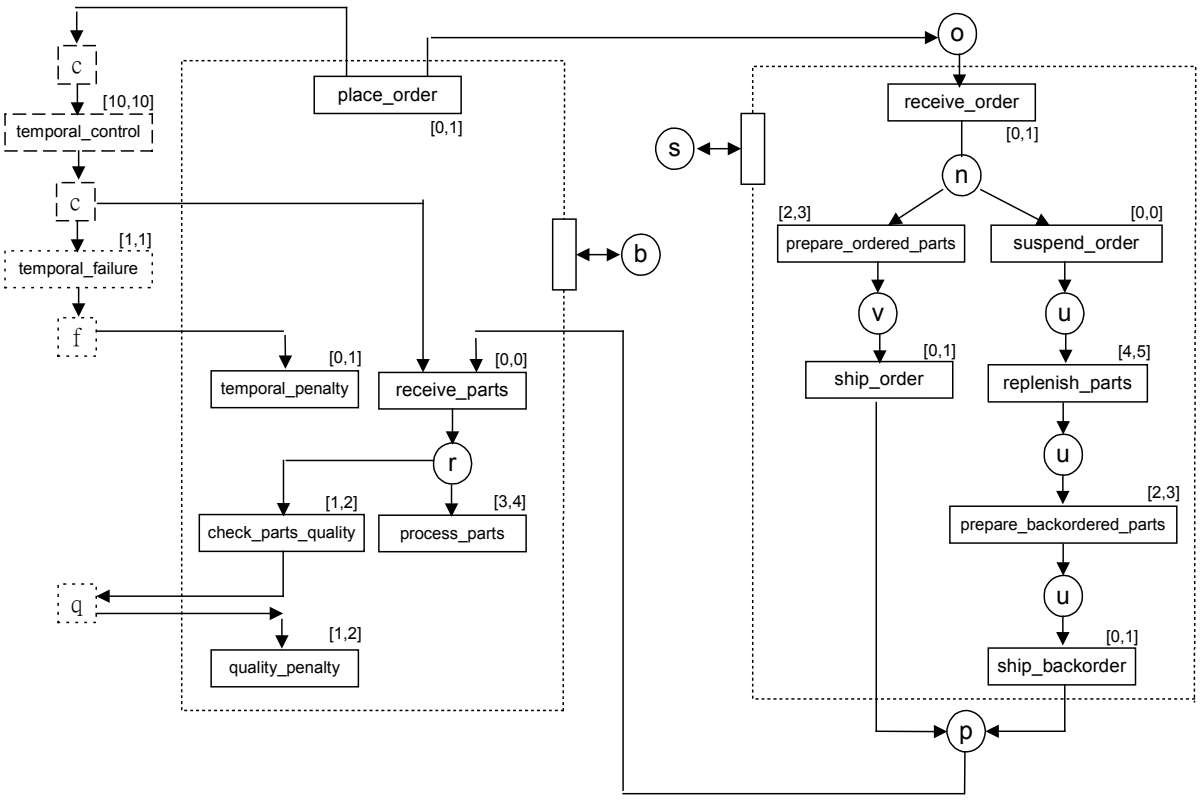


Figure 7: specification of activities that realize capabilities

With this specification, a transition is executable if and only if each of its input places contains an object that together makes its pre-condition true, and since then, its execution occurs within the specified time interval. Once executed, objects in its input places are either consumed or referenced by the transition, and objects in its output places are produced that make its post-condition true. Figure 6 shows the graphical symbols of the activity model.

In Figure 7, an activity model is presented that describes more details how the capabilities in Figure 4 are realized by various activities. As shown in the figure, it can be found that each capability in Figure 4 is mapped here into a corresponding transition except for the process_order( ) capability that is mapped into the prepare_ordered_parts, suspend_order, replenish_parts, and prepare_backordered_parts transitions, and for the ship_order( ) capability that is mapped into the ship_order and ship_backorder transitions. This reflects the fact that the requests/services a role issues may be realized by more working processes in the application. It is also noted that for those transitions derived from the capabilities of buyer and seller roles, they all access (update) the corresponding buyer and seller objects, and this is specified in a succinct manner through respective global access interfaces denoted at the rectangle boxes.

With the activity model based on Petri nets, its formal semantics can then be applied for behavioral verification of its satisfying those rules defined in the commitment model. This can be achieved via decision procedures that traverse the reachability graph derived from the model. The reader is referred to [12] for more details about this issue.

4 The specification method

4.1 Specifying the commitment model

In our method, the specification of collaboration management between VE members starts from describing the desired commitment associated with participant members in a commitment model by using the following steps:

1. Start with describing a desirable collaboration by identifying a commitment object. The

purposes of the collaboration are then explored to identify what roles each member plays in order to fulfill these purposes; with each role identified, a corresponding role object is imposed in the model. Finally, based on the role each member plays, the required requests/services to achieve such a role is defined and specified as the capabilities of the role object.

In our example, as shown in Figure 1, two members participate the collaboration with two roles, buyer and seller, played respectively. For the buyer, it places an order to the seller and expects to receive, check quality of, and process ordered parts. Therefore, the buyer object provides four capabilities as desired. Likewise, the seller object provides three capabilities to achieve its prospect role.

2. With role objects and their capabilities, check with resources that these capabilities may access. As mentioned earlier, such resources represent those things or entities in the application domain where these capabilities may access to achieve their functionality.

   In Figure 1, we can see that the buyer accesses three resources: the order it placed, the shipped parts it expects to receive, and the parts it already received. For the seller, it accesses four resources: the order it expects to receive, the order it received, the ordered parts it prepared, and the ordered parts it shipped.

3. Consider any constraints (e.g., temporal, quality, and cost ones) that need to be complied with during the interaction between capabilities. As a commitment, penalties for violations of these constraints need also be taken into consideration. For this purpose, statements in OTTL are imposed to describe what these constraints are and what penalties for their violations are. If necessary, new capabilities that address the issuing of penalties can be added into appropriate role objects.

   In our example in Figure 1, two (temporal and quality) constraints are identified and specified by the OTTL where their violations are captured explicitly by invoking appropriate penalty capabilities. Therefore, two penalty capabilities are added into the buyer object as desired.

4.2 Specifying the role model

With an initial commitment model, our method advocates the specification of a role model that presents more details how the capabilities behave and how they interact/coordinate with each other under the constraints defined in the commitment model. For the specification of each capability, the following steps are followed first:

1. Identify what resources are required to achieve its functionality, and how it accesses (i.e., references or consumes or produces or updates) these resources. The corresponding objects with respect to these resources become its interaction objects.

2. Determine its pre-condition that is necessarily true for its execution by referencing or consuming input interaction objects.

3. Determine its post-condition that must be satisfied after its execution by producing or updating output interaction objects.

(Note that the time interval of the capability is yet specified until the following situation has been considered.)

We then consider any control/fault objects and associated operations that need to be imposed at/between capabilities to satisfy the constraints defined in the commitment model. For example, as shown in Figure 4, the left part of the first constraint in Figure 1 results in a control object to be produced by the place_order( ) capability and then, after a 10 days delay modeled by a simulated delay( ) operation of the control object, consumed by the receive_parts( ) capability. Also, for satisfying the right part of the constraint: a temporal penalty will be issued within 11 to 12 days, a fault object will be produced by the temporal_failure( ) operation 1 day later after the delay( ) operation has been executed, and then, used to invoke the temporal_penalty( ) capability.

After specifying control/fault objects and their

associated operations, we continue to specify lastly the time interval of each capability. This can be done by checking the constraints identified earlier such that the time interval is specified to satisfy (timing parts of) these constraints. For example, as shown in Figure 4, in order to satisfy the first constraint in Figure 1, the temporal_penalty( ) capability has [0,1] time interval such that, after the 10 days delay imposed by the delay() operation plus the 1 day execution by the temporal_failure( ) operation, its execution represents a temporal penalty issued within 11 to 12 days after an order is placed by the execution of the place_order( ) capability.

4.3 Specifying the activity model

With a role model, the specification of an activity model is then considered that presents more details how the capabilities of each role are realized by a set of working activities where each activity is defined as a task that contributes to the fulfillment of a capability (or part of it). This reflects our view that the capabilities (functionalities) of a role are possibly realized by more working processes in the application. Here, in addition to the trivial case that a capability is realized by a single activity, for a complex capability, its mapping into a set of activities can be done by the following steps:

1. Using the use-case approach [8] to document with a flow of events how the capability is achieved by a sequence of working processes.

2. For each event identified, explore what resources it may access and how it accesses (i.e., references or consumes or produces or updates) them, and determine its pre/post-condition that is necessarily true/satisfied for its occurrence by accessing these resources.

3. For the event first/last in the flow, check if its pre/post-condition is consistent with that of the capability.

4. For each event, identify its time interval such that its execution will occur within the time interval once its pre-condition becomes true. From the first to the last event, however, the lower- and higher-bound summations of their time intervals cannot exceed those of the time interval of the capability.

As shown in Figure 7 that is derived from Figure 4, the process_order( ) capability is realized by the prepare_ordered_parts, suspend_order, replenish_parts, and prepare_backordered_parts activities, and the ship_order( ) capability is realized by the ship_order and ship_backorder activities.

5 Conclusions

Software requirements specification is a key activity in developing a computer-based application. Motivated by the problems in other methods, object-oriented specification methods are developed in order to produce software more understandable and maintainable. The method proposed in this paper is based on the object-oriented paradigm for formal specification of collaboration management in VE. In order to deal with the complexity of modeling contractual collaboration between VE members, commitments, roles, and activities are identified and specified in a top-down fashion. As results, a higher-level commitment model is created first that describes effectively the joined roles and associated rules of a commitment without considering detailed specification. That is, the detailed specification via role and activity models start after all roles and rules have been described in an abstract level. We think this provides better understanding about collaboration management before proceeding too early to formally specify them using some complex notations. Finally, due to its formal semantics of the activity model, behavioral verification of satisfying desired rules can be conducted via formal analysis of the model.

The work for collaboration management in VE is not a new idea. Many researches about it have been done, but none of them provides a complete mechanism for both commitments and visual formalisms for behavioral collaboration. Our method presented herein provides an effort on this issue. As VE gets more attentions in business enterprises, a software system that realizes it becomes now much more desirable. Thus, the development of such a system is a desired field. In our knowledge, using object-oriented techniques together with sound modeling constructs is a promising approach for an effective construction of the system.

As our future work, a tool to facilitate practical application of our model will be constructed. These include a design environment for building the abstract commitment model and then deriving the role and activity models. The specification method presented in section 4 will be integrated with the tool when constructing the three models.

Acknowledgments

The material presented in this paper is based on work supported by Chaoyang University of Technology under Grant CYUT 91-M-004. The authors are grateful to K.Y. Chang and C.Y. Chen for their helps in preparing the artwork of this paper.

References

[1] Agent Builder Environment: http://www.networking.ibm.com/iag/iagsoft.htm.

[2] G. Alonso, et al., "Wise: Business to Business E-Commerce," in Proc. of 9th International Workshop on Research Issues on Data Engineering, IEEE Computer Society Press, 1999, pp. 132-139.

[3] M. Aparicio, IV, et al., "Agent Information Contracts within Virtual Private Networks," in Proc. of 3rd IEEE International Conference on High-Assurance Systems Engineering Symposium, 1998, pp. 304-311.

[4] D. Baker, et al., "Providing Customized Process and Situation Awareness in The Collaboration Management Infrastructure," in Proc. of 4th IFCIS Conf. on Cooperative Information Systems, IEEE Computer Society Press, 1999, pp. 79-91.

[5] D. Georgakopoulos, et al., "Managing Process and Service Fusion in Virtual Enterprises," International Journal of Information Systems, vol. 24, no. 6, 1999, pp. 429-456.

[6] A. Geppert, et al., "Market-Based Workflow Management," Journal of Cooperative Information Systems, vol. 7, no. 4, 1998, pp. 297-314.

[7] Y. Hoffner, "Supporting Contract Match-Making," in Proc. of 9th International Workshop on Research Issues on Data Engineering, IEEE Computer Society Press, 1999, pp. 64-71.

[8] I. Jacobson, Object-Oriented Software Engineering – A Use Case Driven Approach, Addison Wesley, 1992.

[9] K. Jain, et al., "Agents for Process Coherence in Virtual Enterprises," Communications of the ACM, vol. 42, no. 3, March 1999, pp. 62-69.

[10] K. Jain and M. Singh, "Using Spheres of Commitment to Support Virtual Enterprises," in Proc. of 4th ISPE International Conference on Concurrent Engineering: Research and Applications (CE), International Society for Productivity Enhancements (ISPE), Aug. 1997, pp. 469-476.

[11] J. Klingemann, et al., "Deriving Service Models in Cross-Organizational Workflows," in Proc. of 9th International Workshop on Research Issues on Data Engineering, IEEE Computer Society Press, 1999, pp. 100-107.

[12] J. Lin, et al., "Object-Oriented Specification and Formal Verification of Real-Time Systems," Annals of Software Engineering, 1996, vol. 2, pp. 161-198.

[13] H. Ludwig and K. Whittingham, "Virtual Enterprise Coordinator – Agreement-Driven Gateways for Cross-Organizational Workflow Management," in Proc. of International Joint Conference on Work Activities Coordination and Collaboration, ACM Press, 1999, pp. 29-38.

[14] A. Ngu, "Specification of Cooperative Constraints in Virtual Enterprise Workflow," 9th IEEE International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, 1999, pp. 140-147.

[15] J. Peterson, "Petri Nets," ACM Computer Surveys, vol. 9, no. 3, Sep. 1977, pp. 223-252.

[16] J. Peterson, Petri Net Theory and The Modeling of Systems, Prentice- Hall, 1981.

[17] H. Schuster, et al., "The Collaboration Management Infrastructure," 16th IEEE International Conference on Data Engineering, 2000, pp. 677-678.

[18] A. Tarhan, et al., “A Distributed Tool for Commitment Specification and Management,” in Proc. of 25th EUROMICRO Conference, 1999, pp. 210-217.

[19] Y. Wand and R. Weber, “A Model of Systems Decomposition,” in Proc. Of Int’l Conf. on Information Systems, Boston, Dec. 4-6, 1989, pp. 41-51.

[20] X. Yang, et al., “Constraints for information cooperation in virtual enterprise systems,” 3rd International Symposium on Cooperative Database Systems for Advanced Applications, 2001, pp. 159-166.

[21] E. Yiannis and L. Thomas, “Specification and Analysis of Parallel/Distributed Software and Systems by Petri Nets with Transition Enabling Function,” IEEE Transaction on Software Engineering, vol. 18, no. 3, March 1992, pp. 252-261.