◈ Research Paper

Collaborative Object-Oriented Analysis for Production Control Systems

Chang-Ouk Kim[1)]

Abstract

Impact of business process re-engineering requires the fundamental rethinking of how information systems are analyzed and designed. It is no longer sufficient to establish a monolithic system for fixed business environments. Information systems must be adaptive in nature. This demand is also applied in production domain. Enabling concept for the adaptive information system is reusability. This paper presents a new object-oriented analysis process for creating such reusable software components in production domain, especially for production planning and scheduling. Our process called MeCOMA is based on three meta-models: physical object meta-model, data object meta-model, and activity object meta-model. After the three meta-models are extended independently for a given production system, they are collaboratively integrated on the basis of integration pattern. The main advantages of MeCOMA are (1) to reduce software development time and (2) to consistently build reusable production software components.

1. Introduction

In the past half century, manufacturing enterprises used to consider only the quality of product. As the customer demands become more diverse and volatile, however, competitive edge in the world manufacturing market goes to the enterprises that are able to deliver better products to customers faster and cheaper. Accordingly, in order to survive in the world market, manufacturing enterprises should now consider their business processes as the same terms as their products. This new principle has often driven today's manufacturing enterprises to radically reinvent their business processes. Virtually, it has been recognized that this radical reinvention is not practical because its impact is prone to severely disturb the traditional collaboration rules of task processing in organization, resulting in chaotic organization behaviors. Rather, it seems to be safe and economic that the re-engineering is undertaken on a continuous base so that the staffs are gradually harmonized within the new business processes

1) 명지대학교 산업시스템공학부

Demand of continuous business process re-engineering requires the fundamental rethinking of how information systems are analyzed and designed. It is no longer sufficient to establish a monolithic system for fixed business environments. Information systems which support business processes must be adaptive in nature. One of enabling concepts for the adaptive information system is reusability. Since reusable software components [Jacobson *et al.* 1997] are able to support plug-in-play like hardware chips, they make it possible for the information system to accommodate the changes of business processes without much modification.

This paper presents a new analysis process for creating such reusable software components in production domain, especially for production planning and scheduling field. Our process is called MeCOMA (Meta-Model Driven Collaborative Object Modeling Approach). To our knowledge, little efforts have been given to object-oriented modeling of high-level production domain, compared with research works toward object-oriented modeling of shop-floor control.

Unlike conventional object-oriented methodologies, the MeCOMA approach exploits a collaboration scheme of three meta-models: physical object meta-model, data object meta-model, and activity object meta-model. It is a mixed approach of top-down method and bottom-up method. The three meta-models are refined independently for a target production system, after which they are collaboratively integrated on the basis of integration pattern. The idea of MeCOMA is somewhat similar to IDEF0. Primitive function building block is given in IDEF0 and it is instantiated for a given system of modeling. All function blocks with their order in terms of input-output data relationships represent business processes. Compared with IDEF0, MeCOMA uses the three building models (blocks). The main advantages of the MeCOMA approach are (1) to reduce the software development time and (2) to consistently build reusable production software components.

The remainder of this paper is organized as follows: Section 2 discusses the features of production domain and describes a case study concerning production planning and scheduling. Section 3 presents previous research works related to software modeling of production information system, especially focusing on object-oriented methods. Section 4 introduces the MeCOMA approach with its application to the case study. Finally, conclusions and future research works are discussed in Section 5.

2. Description of Production Domain

2.1 Characteristics

As far as production system is concerned, two distinguished modeling aspects are discovered, compared with business systems such as banks and insurance companies. One is hierarchically recursive structure and the other is coordination task. Particularly, recursive structure can be observed in both physical structure and logical structure in production control domain.

From the viewpoint of object-orientation, physical production system can be regarded as a recursive structure consisting of layered resources from company, through factories, shops, and cells, to machines. Figure 1-a describes the object-oriented model of the

physical structure using composite pattern [Gamma *et al.* 1994] where constraints are specified inside curly bracket. The role of resource is the processing of jobs assigned to the resource. Due to the complexity of planning and scheduling, job at company level is also repetitively decomposed as shown in Figure 1-b. Furthermore, Bill-Of-Material (BOM) of end-product has tree form depicted in Figure 1-c.

To make it difficult for modeling this kind of system is that the three recursive structures interact with each other, not being able to analyze them separately. With their coordination behaviors, planning and scheduling activities are carried out. To date, systems that are characterized as the multiple recursive structures with their interactions have not been attempted to model in object-oriented way.

2.2 Case study

In this subsection, we will briefly present a real production system. Throughout this paper, this production system will be used to explain the three meta-model elements of MeCOMA.

This system has four physical layers one company, one factory, two shops, and dozens of machines in each shop. This system manufactures several end-products, parts of automobiles. From automobile enterprises, the company receives product jobs. Manufacturing plan for each product is prepared using Master Planning Schedule (MPS) algorithm. At factory level, each product plan is exploded to generate part plans using Material Requirement Planning (MRP). Parts are scheduled using job shop scheduling algorithm at each shop according to their process plans, .

3. Previous Research Works

Depending on what aspect is emphasized, system analysis methods can be classified into data-oriented methods, function-oriented methods, and object-oriented methods. It is data entities with their relationships that data-oriented methods create for establishing production information systems. Entity-Relationship diagram [Chen 1976], Extended Entity-Relationship diagram [Teorey *et al.* 1986], and Data Flow diagram [Batini *et al.* 1992] are practically applied ones. Whereas these techniques are capable of extracting necessary data and delineating their dependency, they cannot systematically describe dynamic aspect such as activities that change the values of interrelated data entities. To cope with such a weakness, data-oriented methods are usually used coupled with function-oriented methods.

Due to the recursive property of function in nature, most function-oriented methods adopt hierarchical function decomposition. This is also taken into consideration in Structured Analysis and Design Technique method [Ross 1978] that is being widely applied in industry. However, since function-oriented methods do not address the relationships among data entities, they are prone to produce inconsistent, redundant specification of prerequisite input and output data for functions. Furthermore, most methods do not clearly define the level of function decomposition, which is dependent on the skills of analysts. Consequently, these methods degrade the reusability of software components.

For the last decade, object-oriented methods have drawn much attention among software researchers. It has been believed that object-oriented methods are able to guarantee the reusability of software components. In object-oriented modeling, each object type encapsulates associated data and activities (operations). Representative object-oriented methods are OMT method [Rumbaugh *et al.* 1991], Booch method [Booch 1994], Responsibility-driven method [Wirfs-Brock *et al.* 1990], OBA method [Rubin and Goldberg 1992], Objectory method [Jacobson *et al.* 1992], and so on. Generally, these methods present two analysis tools: object description language and development process.

The object description languages define necessary graphical symbols and semantics for expressing object models, such as static and dynamic diagrams. When a target system is given, the development processes suggest systematic building procedures of the object models. Currently, although several object description languages have been exhibited, UML proposed by Rational company is emerging as industry standard.

Broadly, there are two types of object-oriented development processes in commercial software market: object-centered approach and activity-centered approach. In the first approach, defining object types and constructing static diagrams are the primary modeling tasks in the object-centered approach. Activities specified as the interactions of objects are derived from the static diagrams. It seems that OMT method, Booch method, and Responsibility-driven method are typical ones of object-centered approach, although they do not explicitly specify this process order. On the other hand, identifying business processes (sequence of activities) is the first step in activity-centered approach. Object types with static and diagrams are established on the basis of the process description. Typical methods are OBA method and Objectory method. In particular, Objectory introduces the concept of use cases and actors to model processes. The concept of use case is actually a function-oriented one, yet it is an important and widely practiced modeling concept in object-oriented development process.

Objectory is an iterative, incremental approach whereby system requirements are gradually found using use cases. Static and dynamic diagrams are then specified using the use cases[Larman 1998]. Although, Objectory is being popularly applied in industry, it is too general development process tool to consistently model target systems. That is, the analysts are liable to view the same system differently, leading to create different use cases, static diagrams, dynamic diagrams, and to finally produce inconsistent software components. In order to create reusable components enabling plug-in-play, it is essential to suggest a more concrete development process.

In the context of manufacturing, most object-oriented models have been limited to shop-floor control systems [Mize *et al.* 1992, Adiga 1993, Nof 1994, Doscher and Hodges 1997]. CIM-OSA [Jorysz and Vernadat 1990] is a conceptual model that deals with whole manufacturing enterprise. However, the modeling concepts in CIM-OSA are also too general to apply for production systems.

This research proposes a concrete object-oriented development process for production domain. Each of the three meta-models includes invariant conceptual elements with their relationships. For a given production system, they are extended in parallel and collaboratively integrated. Since MeCOMA provides rigorous collaboration pattern, consistent models can be derived.

4. The MeCOMA Model

4.1 Overview

MeCOMA is a meta-model driven process for analyzing the family of production systems in object-oriented way. Here, meta-model is defined as a set of invariant elements with their relationships. MeCOMA is defined as four-tuple:

MeCOMA = { PM, DM, AM, CP }
where PM physical object meta-model
DM - data object meta-model
AM - activity object meta-model
CP - collaboration procedure

PM delineates the types of essential real world entities with their relationships, DM classifies the types of production data, and AM suggests a systematic way of finding production activities. CP expresses the collaborative integration pattern of the three meta-models (See Figure 2-a). The architecture of MeCOMA development process is illustrated in Figure 2-b. Consistency is maintained rigorously by CP, enabling to create reusable production components.

From the next three subsections, the three meta-models are introduced.

4.2 Physical object meta-model

In this subsection, we will introduce the ontological definition of essential objects tangible in production domain and their relationships required for production planning and scheduling.

According to dictionary, system is defined as a group of components working together to achieve goals. In the discourse of business organizations, types of components can be categorized as **processing entity**, **processed entity**, and **coordinator**. The goal of this kind of system is to properly engineer the processed entities by processing entities, while satisfying system constraints. Of the three primary concepts, the roles of coordinators are particularly important because they control over the behaviors of processing entities and processed entities. For each domain, each concept can be specialized.

In the context of production domain, processing units are **resources** such as companies, factories, shops, cells, and machines, while processed units are **items, inventory, resource jobs**, and **time-phased jobs**. The relationships among these objects are shown in Figure 3-a. Hereafter, the system consisting of a resource and associated processed entities are called **physical system**, which is modeled using Package diagram suggested by UML [Eriksson and Penker 1998]. Package means a group of semantically related objects.

A resource job is a cluster of an item that is assigned to the resource for manufacturing. Ordered products assigned to company and scheduling jobs to shop are typical examples of resource jobs. Due to the complexity of planning and scheduling tasks, a resource job is usually divided into several small jobs, each of which is allocated in a fixed time interval. This kind of job is called time-phased jobs. Representative ones are time slot jobs resulting from MPS and those from MRP.

In Figure 3-a, two interfaces denoted as hollow circles are linked to resource and resource jobs, respectively. They provide connection points to the physical system. In other words, either upper-level system or lower-level system can access the system through the interfaces. On the other hand, the system may need to access others. Dependency relation denoted as dotted line with arrow is used for this possibility. Therefore, three recursive structures in production domain discussed in Section 2 can be explained only by the recursive structure of physical system.

One aspect that is not considered in the physical system is the role of coordinator. Virtually, components in real system perform cooperative works under cooperation plans. In production systems, it is the (production) **manager** who prepares such cooperation plans. He/She also keeps track of the progress states of the plans, and adjusts them if the actual processing is behind the plans. To take into account this feature in meta-model, the concept of **control system** is conceived. Figure 3-b shows the control system. It is a logically constructed system embedding a physical system and several sub-control systems.

In the control system, the manager undertakes two important coordination roles one within its physical system and the other between its physical system and its sub-control systems. The first role is that the manager generates time-phased jobs for each resource job in its physical system. The second role is that he/she decomposes the time-phased jobs of its physical system into the resource jobs in the physical systems of the sub-control systems. In general, both of the managers roles are performed with the aid of planning or scheduling algorithms. Knowing that the control system is able to comprise its offspring makes the physical object meta-model be flexibly expanded relying upon the structures of target production systems.

Figure 4 shows the extended physical object model for our case study.

4.3 Data object meta-model

Associated with each physical object are attributes that describe its states and properties. In traditional abstract data approach [Booch 1994], such attributes belong to physical objects as member variables. However, we deliberately partition the attributes into semantically related subsets and regard them as individual data objects, because previous experiences tell us that most business information systems define the type of data object as schema and keep the data objects as instantiated tuples of the schema in databases. Hence, it is necessary to analyze business information system coupled with the database concept. This idea is realized as data object meta-model in MeCOMA. Figure 5 shows the data object meta-model. At the first level, the data object meta-model classifies production data into **property data objects** and **relation data objects**.

Property data objects express the description, constraints and current states of physical objects. For instance, item master data object contains descriptive data of item object, resource data object defines the maximum capacity of resource, inventory data object keeps the current level of storage, and so on.

Relation data describe the coordination plans between physical objects. They are further categorized as **static data objects** and **dynamic data objects**. In production domain, static data objects imply pre-defined coordination constraints. Of typical are BOM data object

that expresses assembly structure among items. Dynamic data objects are coordination data between physical objects that change frequently by the manager. MPS, MRP, and shop scheduling data are representative ones.

Figure 6 shows the data objects for our case study.

4.4 Activity object meta-model

Discovering and representing user requirements are the most important step in the analysis and design of information system, whether it is object-oriented system or function-oriented system. Use case approach proposed in UML is popular one for this purpose. This approach does not view use case as an object. Instead, a use case is a narrative, textual description of the sequence of events and activities whereby physical objects with their relationships are derived. Compared with the use case approach, MeCOMA intentionally treats activity as an object in the sense that everything that can be conceptualized is object. This fact is reflected in activity object meta-model.

It is well recognized that a single physical object rarely accomplishes an activity. Accordingly, the issues concerning activity modeling in object-oriented method are how each activity can be split into the operations of objects, how the operations are controlled and who is responsible for the control. Unfortunately, there is no systematic, consistent method to define activities and to divide them into the operations of object yet.

In MeCOMA, finding and refining activity is more concrete than the use case approach. As shown in Figure 7-a, an event of interest triggers an activity that can be either **operationalized activity** or **abstracted activity**. The former is defined as an activity that only accesses a property data object or a relation data, while the latter is a high-level goal that should be refined until it is materialized by the collaboration of operationalized activities. Unlike function-oriented approach, this approach suggests a definite stopping criterion for the activity decomposition. That is, the decomposition is stopped when abstracted activities are realized by operationalized activities. As a result, abstracted activities generated during the decomposition procedure become software components.

Operationalized activities are classified into three types: **singular activity**, **social activity**, and **coordination activity**. Singular activity manipulates the property data object of a physical object and is called the operation of the object. Social activity is concerned with behavioral rule associated with human beings. Authorization and confirmation by manager belong to social activities. Coordination activity controls the cooperation of physical objects, so handles dynamic data object.

While the activity decomposition step is being progressed, relationships among activities are created. The types of such relationships are depicted in Figure 7-b. Between parent activity and children activities, there exist **And** refinement relation denoted as dark circle and **Or** refinement relation denoted as hollow circle. Among sibling activities, **Parallel** and **Precede** relations exist. These relationships are dependent on the requirements of production system being explored.

Figure 8 depicts the extended activity object model for our case study.

4.5 Collaboration procedure

In this subsection, we present an integration methodology of the three meta-models. Conventional object-oriented methods primarily focus on physical objects and then model data and activities around the objects. On the other hand, MeCOMA puts the same emphasis on the three meta-models. This is materialized through their parallel extensions. Nevertheless, it is necessary to integrate them because they are tightly coupled activities are done by the cooperation works between physical objects, which are reflected in the data objects related to the physical objects. This fact is illustrated in integration pattern shown in Figure 9.

Associated with each physical object are property data. Between physical objects, there exists a relation data object that specifies their cooperation behavior. According to the type of the relation data object, the cooperation between physical objects is dynamically changed (dynamic data object) or fixed (static data object). Singular activity manipulates a property data object and so is performed by the corresponding physical object. Namely, it will become one of the operations of the physical object at design stage. Manager performs coordination activity and updates associated dynamic data object.

Figure 10 depicts the collaboration process of the three meta-models using sequence diagram defined in UML. The meta-models can be expanded concurrently, after which they are synchronized based on the integration pattern. In the collaboration process, data object should be associated with physical objects using the integration pattern. This is also applied between activity object meta-model and data object meta model. The activity object meta-model requests input and/or output data to the data object meta-model, which is satisfied if such data are declared in the data object meta-model. If not, categorize the data in the data object meta-model and ask the physical object meta-model of associating the data with a physical object or between physical objects. If the physical object meta-model cannot find such object, it should be further unfolded.

Figure 11 shows part of collaboration procedure for our case study. After parallel extensions of the three meta-models, activity object meta-model requests input and output data of inventory assignment activity to data object meta-model. In this situation, assume that assigned inventory data was not found. Data object meta-model then creates the assigned inventory data object and asks physical object meta-model of the position of the data object. Based on the integration pattern, physical object meta-model associates the data object between resource and inventory. The same procedure is applied to other activities.

5. Conclusion

Construction of production information system is one of the most important steps toward truly embodying computerized manufacturing system. Despite, much effort has not been dedicated to production domain by researchers. Throughout this paper, we propose an object-oriented way of building hierarchical production information system. It is meta-models and collaboration process suggested in MeCOMA that are distinguished from other object-oriented methods. We believe that this approach can be used for various types of production system. This is confirmed through an application to real manufacturing

company illustrated in this paper. Currently, other cases are also being tested.

References

[1] Adiga, S., 1993, *Object-Oriented Software for Manufacturing Systems* (Chapman & Hall).

[2] Batini, C., Ceri, S., and Navathe, S. B., 1992, *Conceptual Database Design* (Benjamin/ Cummings Publishing Company, Inc).

[3] Booch, G., 1994, *Object-Oriented Analysis and Design* (Addison-Wesley).

[4] Chen, P., 1976, The entity-relationship model: toward a unified view of data. *ACM Transactions on Database System*, **1**, 9-36.

[5] Doscher, D. and Hodges, R., 1997, SEMATECHs experiences with the CIM framework. *Communications of the ACM*, **40**(10), 82-84.

[6] Eriksson, H.-E. and Penker M., 1998, *UML Toolkit* (John Wiley & Sons).

[7] Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994, *Design Patterns Elements of Reusable Object-Oriented Software* (Addison-Wesley).

[8] Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G., 1992, *Object-Oriented Software Engineering* (Addison-Wesley).

[9] Jacobson, I., Griss, M., Jonsson, P., 1997, *Software Reuse: Architecture, Process and Organization for Business Success* (Addison-Wesley).

[10] Jorysz, H. R. and Vernadat, F. B., 1990, CIM-OSA part 1: total enterprise modelling and function view. *International Journal of Computer Integrated Manufacturing*, **3**, 144-156.

[11] Jorysz, H. R. and Vernadat, F. B., 1990, CIM-OSA part 2: information view. *International Journal of Computer Integrated Manufacturing*, **3**, 157-167.

[12] Larman, C., 1998, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design* (Prentice-Hall).

[13] Mize, J. H., Bhuskute, H. C., Pratt, D. B., and Kamath, M., 1992, Modeling of integrated manufacturing systems using an object-oriented approach. *IIE Transactions*, **24**(3), 14-26.

[14] Nof, S. Y., 1994, Critiquing the use of object-orientation in manufacturing. *International Journal of Computer Integrated Manufacturing*, **7**, 3-16.

[15] Ross, D., 1977, Structured analysis (SA): a language for communicating ideas. *IEEE Transactions on Software Engineering*, **3**(1), 16-34.

[16] Rubin, K. and Goldberg, A., 1992, Object behavior analysis. *Communications of the ACM*, **35**(9), 48-62.

[17] Rumbaugh, J., Blaha, M., Eddy, F., and Lorensen, W., 1991, *Object-Oriented Modeling and Design* (Prentice Hall).

[18] Teorey, T. J., Yang, D., and Fry, J. P., 1986, A logical design methodology for relational databases using the extended entity relationship model. *ACM Computing Survey*, **18**, 197-222.

[19] Wirfs-Brock, R., Wilkerson, B., and Wiener, L., 1990, *Designing Object-Oriented Software* (Prentice-Hall).

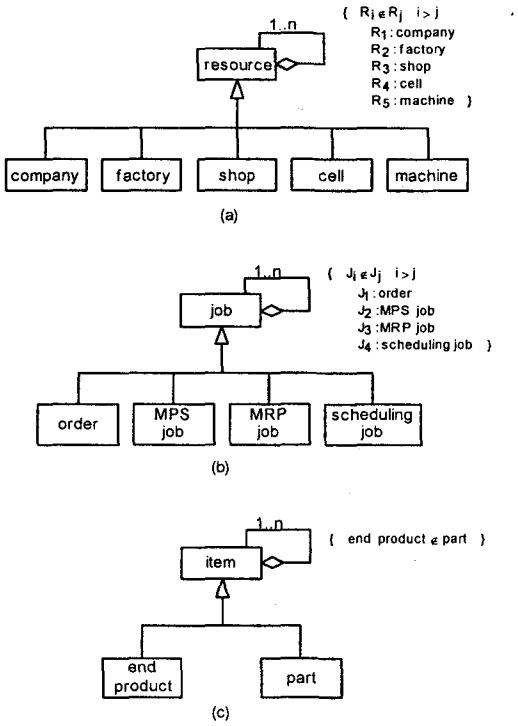

Figure 1. Recursive structure of (a) resources, (b) jobs, and (c) items

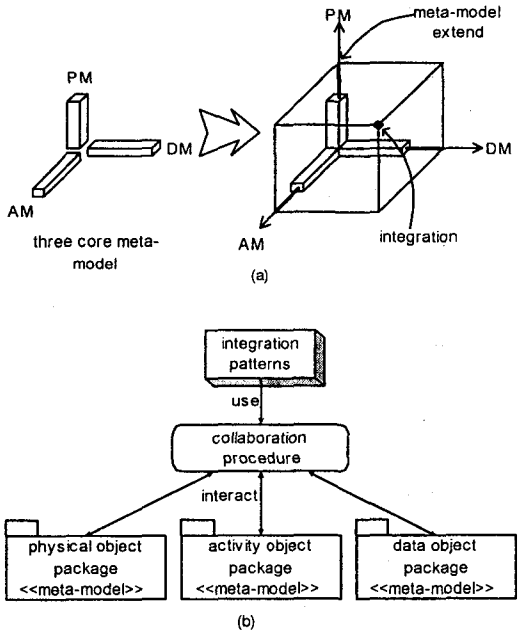

Figure 2. MeCOMA development process (a) concurrent meta-model extension and (b) architecture of development process

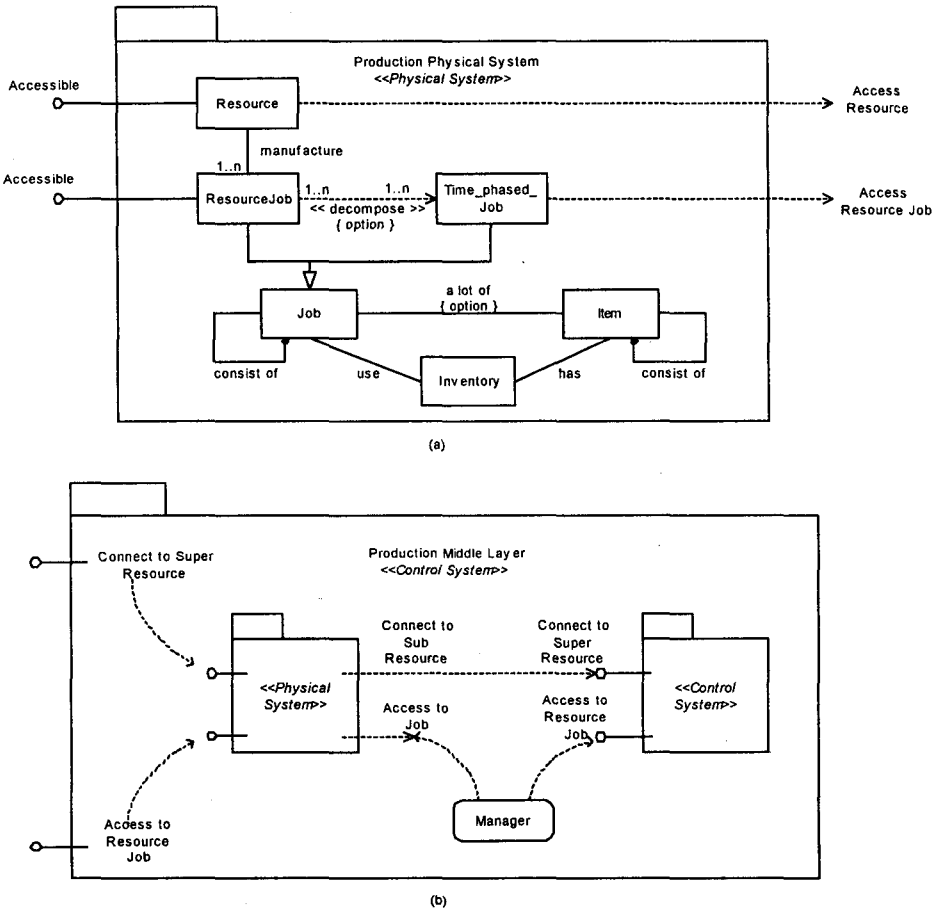

Figure 3. Physical object meta-model (a) physical system package and (b) control system package

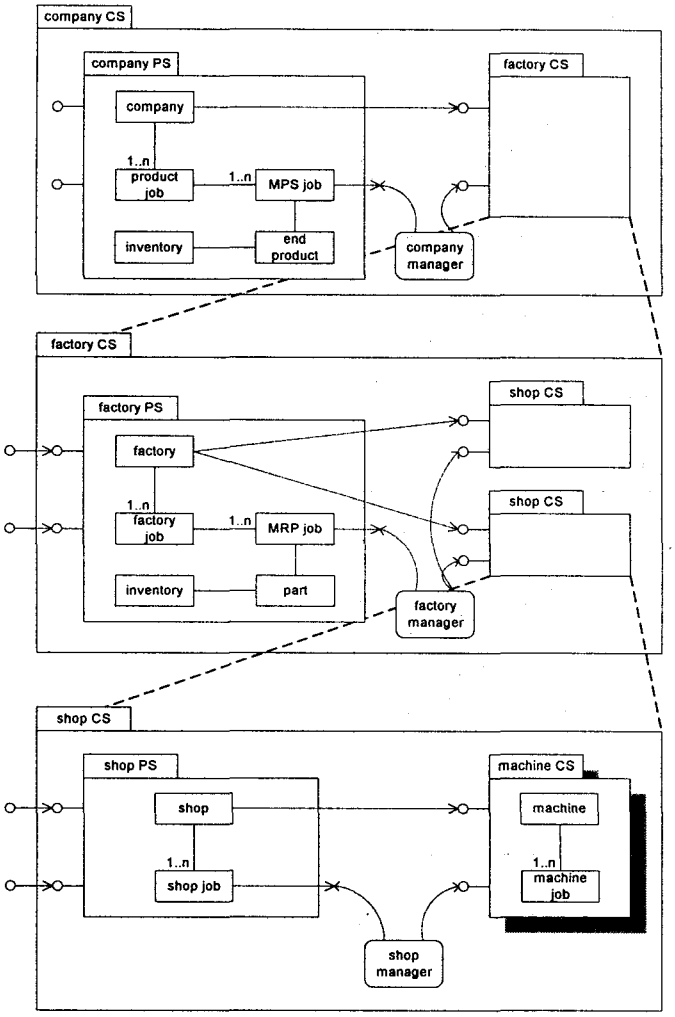

Figure 4. Extended physical object model for case study

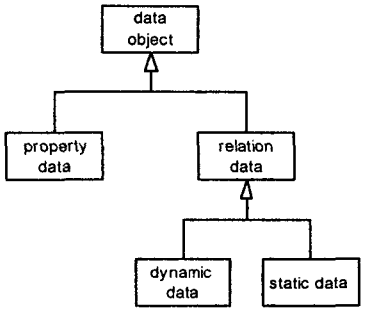

Figure 5. Data object meta-model

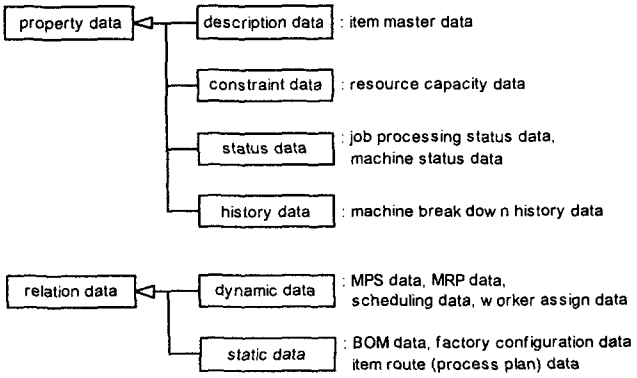

Figure 6. Extended data object model for case study

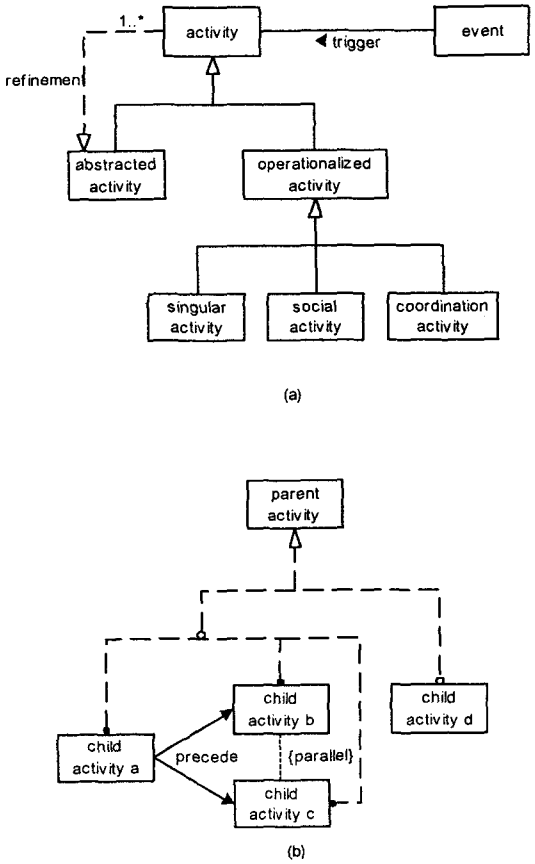

Figure 7. Activity object meta-model (a) activity object hierarchy and (b) activity relationship

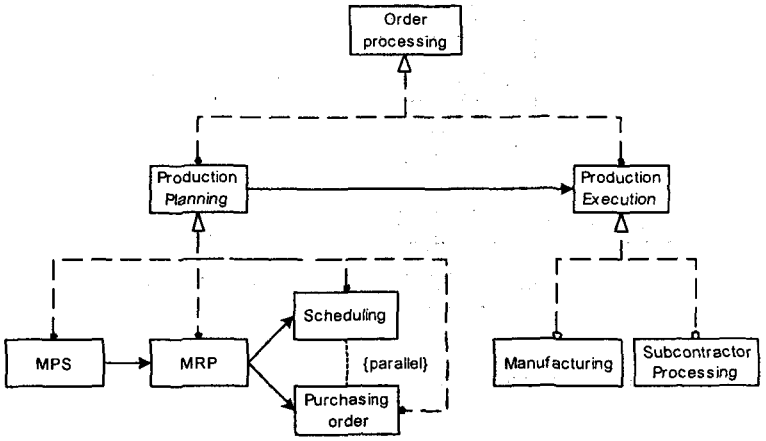

Figure 8. Extended activity object model for case study

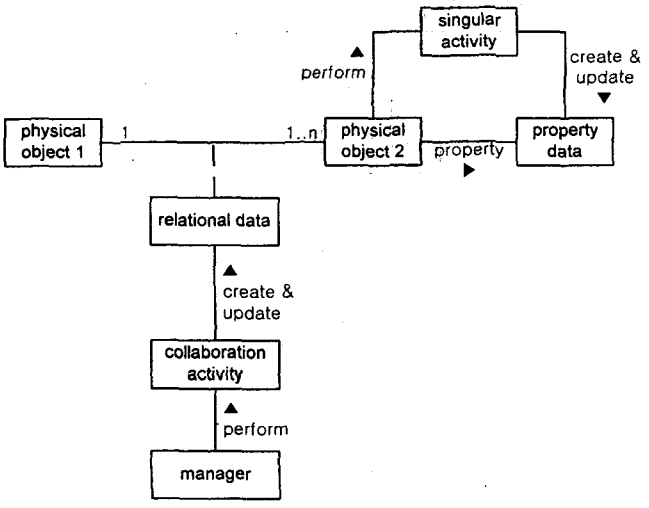

Figure 9. Collaboration procedure: integration pattern

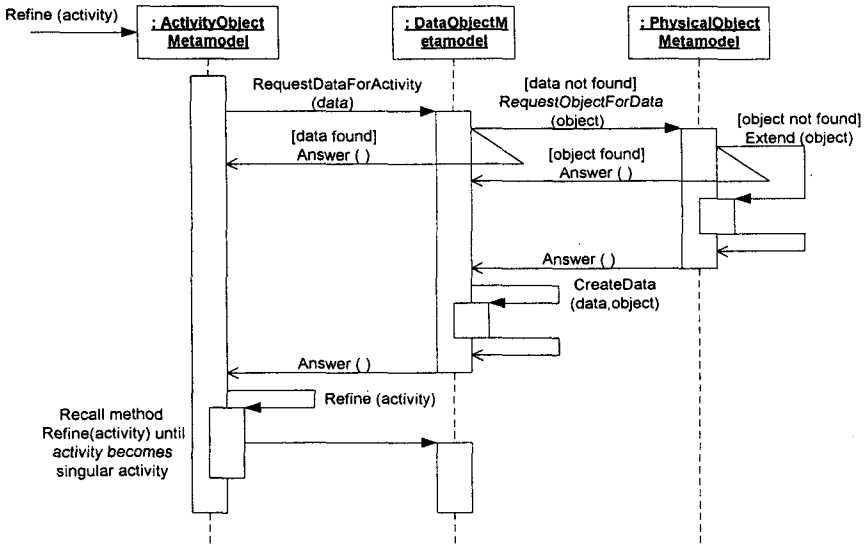

Figure 10. Collaboration procedure: collaboration process

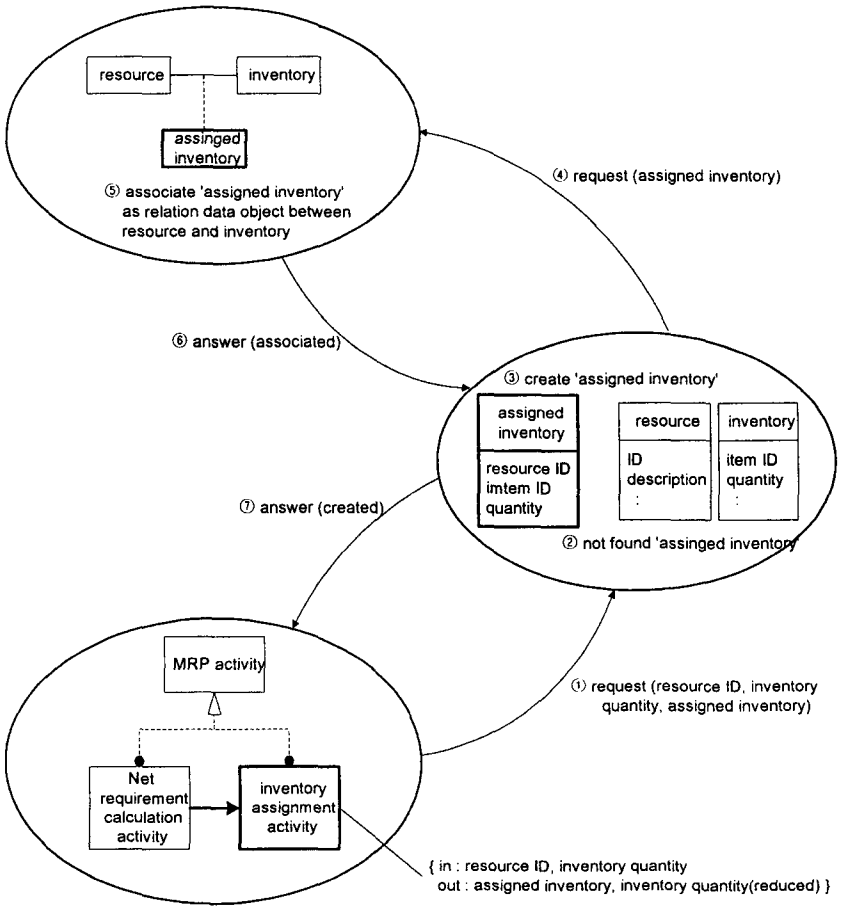

Figure 11. Collaboration procedure for case study