Available online at www.sciencedirect.com

SCIENCE DIRECT®

Robotics and Computer-Integrated Manufacturing 20 (2004) 439–456

Robotics and Computer-Integrated Manufacturing

www.elsevier.com/locate/rcim

# Developing a distributed knowledge model for knowledge management in collaborative development and implementation of an enterprise system

Cheng-Ter Ho[a], Yuh-Min Chen[b,*], Yuh-Jen Chen[b], Chin-Bin Wang[c]

[a] *Department of Industrial Engineering and Management, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, ROC*
[b] *Institute of Manufacturing Engineering, National Cheng Kung University, Tainan 70101, Taiwan, ROC*
[c] *Department of eCommerce Management, Nan-Hua University, Chia-Yi, Taiwan, ROC*

Received 17 December 2002; received in revised form 14 February 2004; accepted 23 March 2004

**Abstract**

Recently, enterprise systems have been extensively adopted to boost enterprise competitiveness. The development and implementation of enterprise systems is a knowledge intensive procedure, being related to enterprise processes and involving information, system and software engineering technologies. Consequently, knowledge management is required to enhance the effectiveness of enterprise system development and implementation, thus helping to increase industrial competitiveness.

This study aims to develop a distributed knowledge model for knowledge management, capable of supporting the collaborative development and implementation of enterprise systems. This objective can be obtained by performing the following tasks: (1) modeling and characterization of the collaborative development and implementation process, (2) identification, analysis and modeling of involved knowledge, and (3) development of a distributed knowledge model for knowledge management related to the collaborative development and implementation of enterprise systems.
© 2004 Elsevier Ltd. All rights reserved.

*Keywords:* Knowledge management; Enterprise engineering; Enterprise systems

## 1. Introduction

With the development of the knowledge economy, knowledge is set to become the most important asset of enterprises in the 21st century. The successful accumulation and employment of enterprise knowledge thus will be one of the keys to the success of enterprises. Consequently, organized management of knowledge resources to help knowledge workers facilitate the creation, access, storage and application of knowledge, will be the main concern of enterprises in the age of the knowledge economy.

"Enterprise systems" is a broad term that is used here to mean a suite of integrated and enterprise-wide software modules that are used to help manage a company's human resources, financial, and the service and/or manufacturing processes, and increase industry competitiveness [1–4]. The development and implementation of an enterprise system is complicated owing to the close relationship between such systems and business processes, and the involvement of management, engineering, and information technologies.

Recently, the concept of allied concurrent engineering [5–8] has been employed in product and process development across enterprise boundaries to enhance product marketability. This concept is also applicable to the development and implementation of enterprise systems because of it involving various disciplines and distributed development characteristics. Collaborative enterprise system development and implementation is based on a distributed and collaborative process, where individuals involved in different disciplines and

*Corresponding author. Tel.: +886-6-275-7575X63922; fax: +886-6-208-5334.
*E-mail address:* ymchen@mail.ncku.edu.tw (Y.-M. Chen).

0736-5845/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2004.03.006

enterprises cooperate in conducting related activities through remote coordination, communication, management and control.

The development and implementation of enterprise systems is a knowledge intensive process, involving business process analysis, modeling, and re-engineering, system requirement analysis, system customization or development, and system implementation and evaluation. Each of the tasks requires employing and sharing of various aspects of knowledge and experience. Consequently, a knowledge management approach capable of supporting the accumulation, sharing and reuse of knowledge and experience in a distributed development environment is required for effective and successful development and implementation of enterprise systems.

Many studies have focused on knowledge management strategies and methodologies from an organizational perspective [9–14]. Many useful studies on the development of information sharing systems and collaborative tools to support intra-organizational processes and teamwork also exist [5,15–17]. However, both the methodology and software system functionality for knowledge management are highly dependent on the process characteristics and the knowledge involved. Consequently, besides determining the knowledge management strategy and methodology, the following two steps are required to establish a full KM environment: (i) identification of target business processes and their characteristics and (ii) identification, analysis and modeling of involved knowledge.

This study describes the development of a distributed knowledge model capable of supporting knowledge management and sharing in the collaborative development and implementation of an enterprise system. In this context, knowledge management is defined to include all aspects of data management, information sharing, and distribution channels.

## 2. Process characterization and modeling

This section first defines enterprise system development and implementation based on the concept of collaboration. A characteristic analysis of the collaborative process for enterprise system development and implementation is then conducted to pave the way for process modeling. To formally describe the collaborative development and implementation process, a process model is then established using a process modeling technique.

### 2.1. Process definition and characterization

Collaboration refers to informal, cooperative relationships that build the shared vision and understanding needed for conceptualizing cross-functional linkages in the context of knowledge intensive activities. Collaboration facilitate the acquisition and integration of resources through external integration and cooperation with other cooperative or supporting enterprises, conducted on a basis of common consensus, trust, cooperation, and sharing by a multi-functional team of experienced knowledge workers. Collaboration is thus essential in knowledge generation and transfer.

Based on the concept of collaboration, collaborative enterprise system development and implementation is defined as follows:

*A collaborative process* for enterprise system development and implementation is an integrated, distributed and cooperative process, in which individuals involved in different disciplines cooperate in software development and system implementation through remote coordination, communication, and control. The key to successful collaboration is complete understanding and effective sharing of system development and implementation information and knowledge throughout the development cycle.

An enterprise system can be developed and implemented by subcontracting some of the development and implementation activities, and thus this execution relies heavily on:

(1) formation, control, coordination and communication of remote processes.
(2) dynamic and loose integration among implementation activities, application systems and knowledge.
(3) management and sharing of knowledge related to development and implementation activities from various heterogeneous resources and environments.
(4) quick and easy changes in the process itself when a need for change exists based on various discipline capabilities of cooperative enterprises.

### 2.2. Process modeling

By investigating studies on the development and implementation of enterprise systems [15,16,18,19], most of the development and implementation of enterprise systems involves three phases, namely business and process analysis, system development, and system realization.

The business and process analysis phase includes the steps of business analysis, process analysis and modeling, and process re-engineering. Business analysis aims to identify the business vision, organizational structure, and functions of business units to guide the re-engineering process to meet the corporate mission. Similarly, process analysis and modeling are intended to identify the motivation and goals for developing and implementing the enterprise system and to fully understand the characteristics of the business processes, thus facilitating process re-engineering. Process analysis and modeling produce an as-is process model that can be applied to re-engineering. Process re-engineering is conducted by employing principles of concurrent engineering and information technology based on the as-is model. The objectives of process re-engineering are to increase the effectiveness of the process, and ultimately to reach the process goals. Process re-engineering creates a to-be process model.

The system development phase includes the steps of requirement analysis, system planning, system design, and system modeling. The requirement analysis and system planning are defined as what the desired system must do and how it will do this, respectively. Meanwhile, the system design makes high-level decisions regarding the overall framework. Finally, the system modeling describes the structures and behaviors of the elements of the software system and their relationships.

The system realization phase includes the steps of system implementation, integration, testing and improvement. System implementation can be conducted by establishing underlying database systems and coding the software components defined in the system modeling stage. Finally, system improvement is realized through system integration and testing.

Modeling the collaborative process of enterprise system development and implementation requires representing the characteristics and behaviors of the process elements and their relationships. A process is a flow of activities triggered by real-world events and executed by various types of organizational units using tools and methods to process knowledge items.

To properly model the collaborative development and implementation process, an enhanced IDEF0 activity diagram is devised and used as the basic construct for process modeling. The graphical representation of activities is shown in Fig. 1, which employs a box with six arrows. The first "I" stands for an activity input, which is provided by a supplier activity. The second "I" represents interactions between activities. Moreover, the "C" indicates activity constraints. Furthermore, the "O" denotes the output from the activity to a consumer activity. The "R" denotes the resources sourced from a repository and employed to perform the activity. Finally, the "C" indicates the allied enterprise that either subcontracts the activity or performs it in collaboration with prime enterprise.

The enhanced IDEF0 activity diagram was used to model the collaborative process of the development and implementation of enterprise systems, as shown in Fig. 2.

Inputs to an implementation activity may be information or knowledge items that are transformed by the activity. Meanwhile, outputs from such an activity may be the results of information or knowledge items transformed by the inputs of the activity. Activity is performed based on standard methods and policies. An activity may interact with no other activities at all, or may interact with any number of other activities. Similarly, either none or any number of collaborative enterprises may exist for a given activity. The resources required to support an activity include knowledge workers, teams, tools, equipment, or reference knowledge.

The activities involved in the collaborative process for developing or implementing of an enterprise system can be divided into real activities and distributive activities. An activity is real if it is performed by the primary enterprise, and is distributive if it is subcontracted to, or supported by other allied enterprises. Moreover, the distributive activities can be further classified into independent activities, dependent activities, and coupled activities. The independent activities are subcontracted and performed entirely by an allied enterprise without any interaction with real activities. Meanwhile, the dependent activities are subcontracted and performed by an allied enterprise with interactions with real

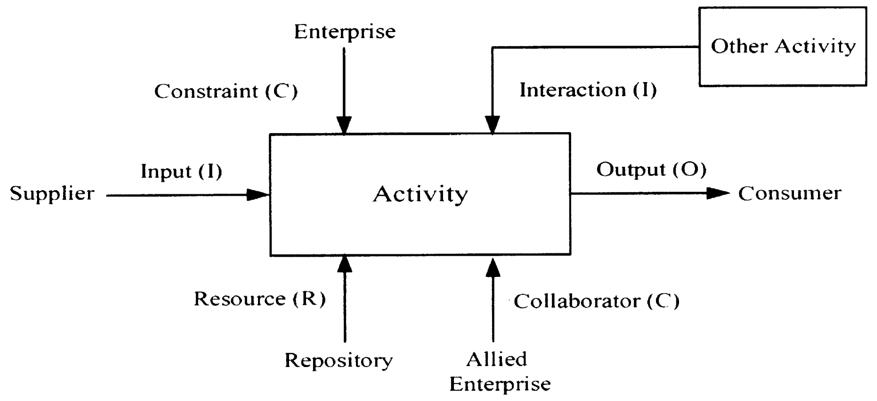


Fig. 1. Activity diagram.

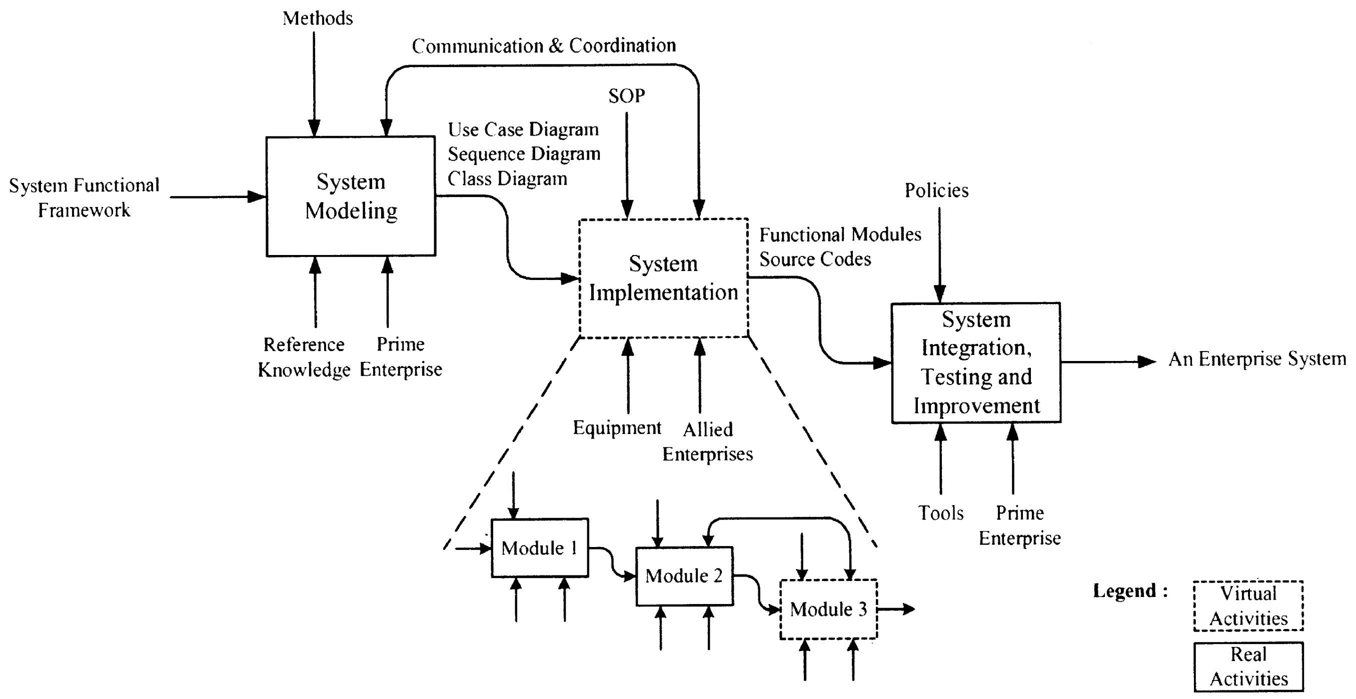


Fig. 2. The collaborative process of enterprise system development and implementation.

activities. The coupled activities are performed cooperatively by the primary enterprise and together with one or more allied enterprises.

The higher-level activities in the collaborative enterprise system development and implementation process can be further decomposed into low-level activities. The hierarchical structure of the process of collaborative enterprise system development and implementation fully reflects its hierarchical, distributed, collaborative, and dynamic natures.

## 3. Knowledge characterization and modeling

This section first identifies the knowledge items involved in the collaborative process of enterprise system development and implementation. Next, the characteristics, attributes, and semantics of the knowledge items, as well as the relationships among the knowledge items are analyzed. Finally, a conceptual model is developed to represent the above findings and the scenario of the knowledge model is described to explain how the knowledge model is to be used.

### *3.1. Identification and classification of knowledge items*

From the perspective of knowledge modeling, the collaborative process for development and implementation of an enterprise system can be viewed as the process of knowledge workers employing resources to perform activities that use, create and transmit knowledge to achieve activity goals under business rules or policies. Therefore, knowledge items involved in collaborative development and implementation process can be identified from the "Inputs", "Outputs", "Constraints", "Resources", and "Interactions" of the enhanced IDEF0-based process model, as well as the "Activities" themselves, as discussed in Section 2. The identified knowledge items are classified as displayed in Fig. 3.

*Inputs* to a particular activity include information or knowledge items provided by one or more preceding activities which are transformed by the activity, while *outputs* of an activity include the results transformed from the inputs by the activity based on activity knowledge. "*Inputs*" and "*Outputs*", including process models, organizational models, functional models, module models, description documents, and software components are essentially the knowledge items flowing along the process and thus are termed "flow knowledge items".

*Constraints* involved in the process include operation standards, methods, policies, and guidelines, which are the explicit knowledge required for the development and implementation of an enterprise system.

*Interactions* involve information and knowledge sharing during interactions among activities. Most interactions involve on-line messages between development team members and transmitted in the form of e-mail, net meetings, and discussions. Besides the on-line messages, posted information is also common in the development process, and essentially involves messages posted in a common area to share information and

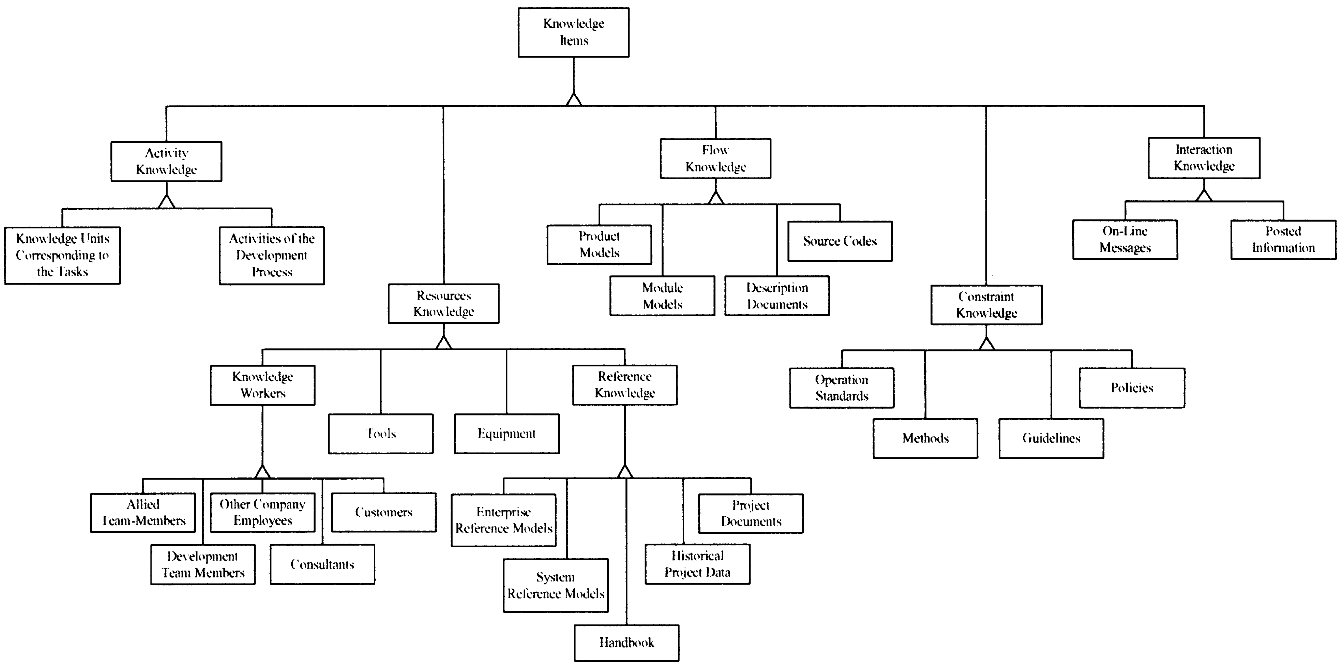


Fig. 3. Classification of knowledge items.

knowledge among team members. Examples of posted knowledge include questions and answers, performance status, and performance results.

*Resources* involved in the process can be categorized into knowledge workers, tools, equipment, and reference knowledge. The knowledge workers include: allied team-members, employees of other companies, customers, development team members, and consultants. These workers act as knowledge creators and consumers. Moreover, activity reference knowledge items serve as references for activity performance. Typical examples of activity knowledge items include enterprise reference models, system reference models, handbooks, historical project data, and project documents.

Most of the above mentioned knowledge items involve explicit knowledge. These items are essentially information items, and can be further classified as involving structured or unstructured information. Structured information items, such as order forms, functional requirement specifications, and so on, can be stored in a commercial database. Meanwhile, unstructured information items, such as enterprise models, project documents, and so on, should be stored in a repository such as a file server.

Unlike explicit knowledge items, activity knowledge is the tacit knowledge that a knowledge worker uses to perform an activity. Most activity knowledge is experience-based. This type of knowledge can be identified using various collection methods, including observation, interviews with experienced workers, and the study of factual documents, handbooks, textbooks, and so on. Activity knowledge can be further classified into knowledge units corresponding to the tasks and activities involved in the development process.

## 3.2. *Knowledge item analysis*

In this section, the main items of knowledge characterization analysis include the attributes and semantics of knowledge items and the relationships among knowledge items.

### 3.2.1. *Relationship analysis*

The relationships among knowledge items can be identified based on the flow model of knowledge items derived from the collaborative development and implementation process model. The main stream for the flow model of knowledge items is defined as the flow of knowledge items along the development process that can be obtained by connecting the inputs and outputs of the activities. The reference knowledge items, constraint knowledge items, and activity knowledge of each activity indicate the flow knowledge items of that activity. Furthermore, knowledge related to interaction between different activities also indicates the flow knowledge items of different activities simultaneously. They together form the flow model of knowledge items as shown in Fig. 4.

The creation of any flow knowledge item not only requires knowledge on constraints, references, and interaction, but also depends on activity knowledge. Activity knowledge is formed through design intent, which applies design rules derived from the activities knowledge base, as shown in Fig. 5.

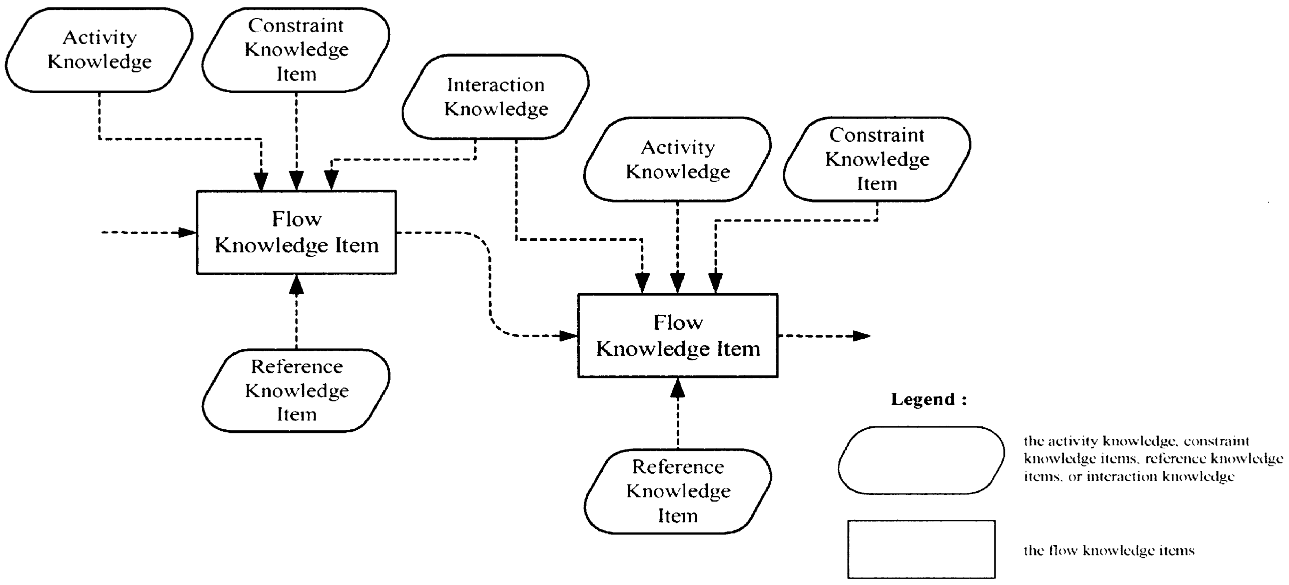


Fig. 4. Flow model of knowledge items.

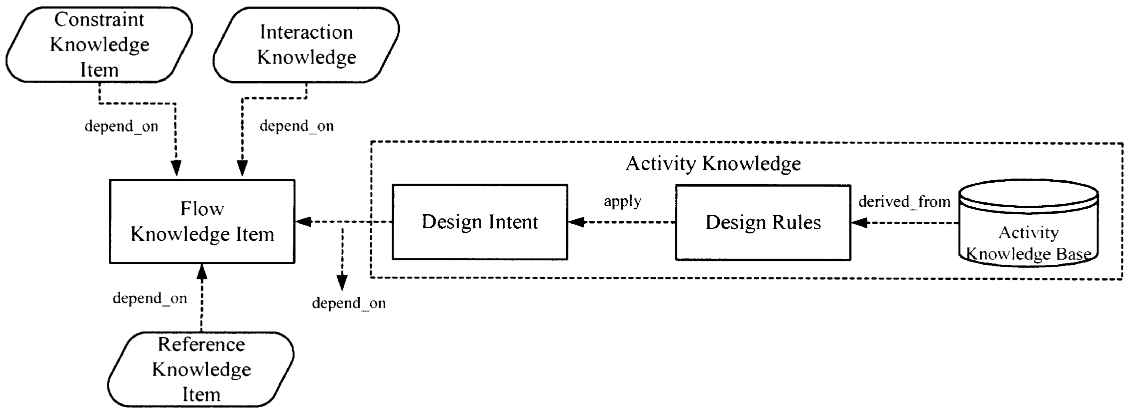


Fig. 5. Factors use to create flow knowledge items.

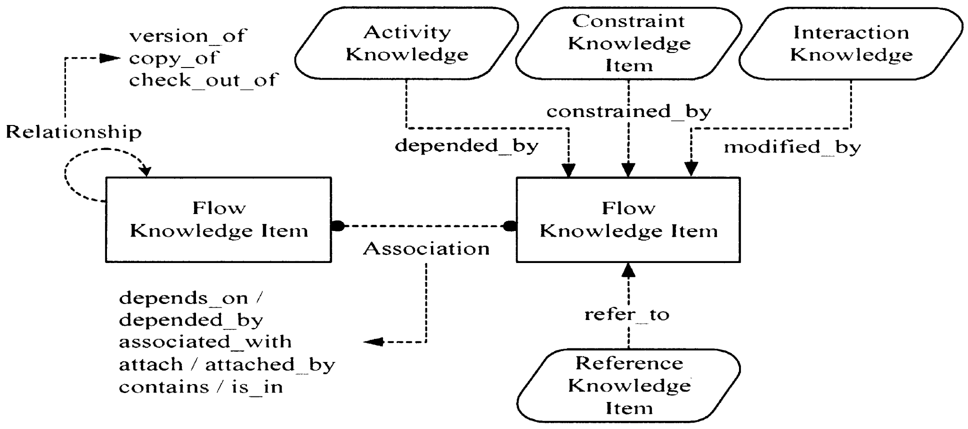


Fig. 6. Types of relationships existing between knowledge items.

From the above analysis, relationships between knowledge items can be classified as (1) relationships between flow knowledge items, (2) relationships between flow knowledge items and reference knowledge items/ constraint knowledge items, (3) relationships between knowledge items and their duplications, (4) relationships between flow knowledge item and activity knowledge, and (5) relationships between flow knowledge items and interaction knowledge (see Fig. 6). The first type of relationships include depends_on/depended_by,

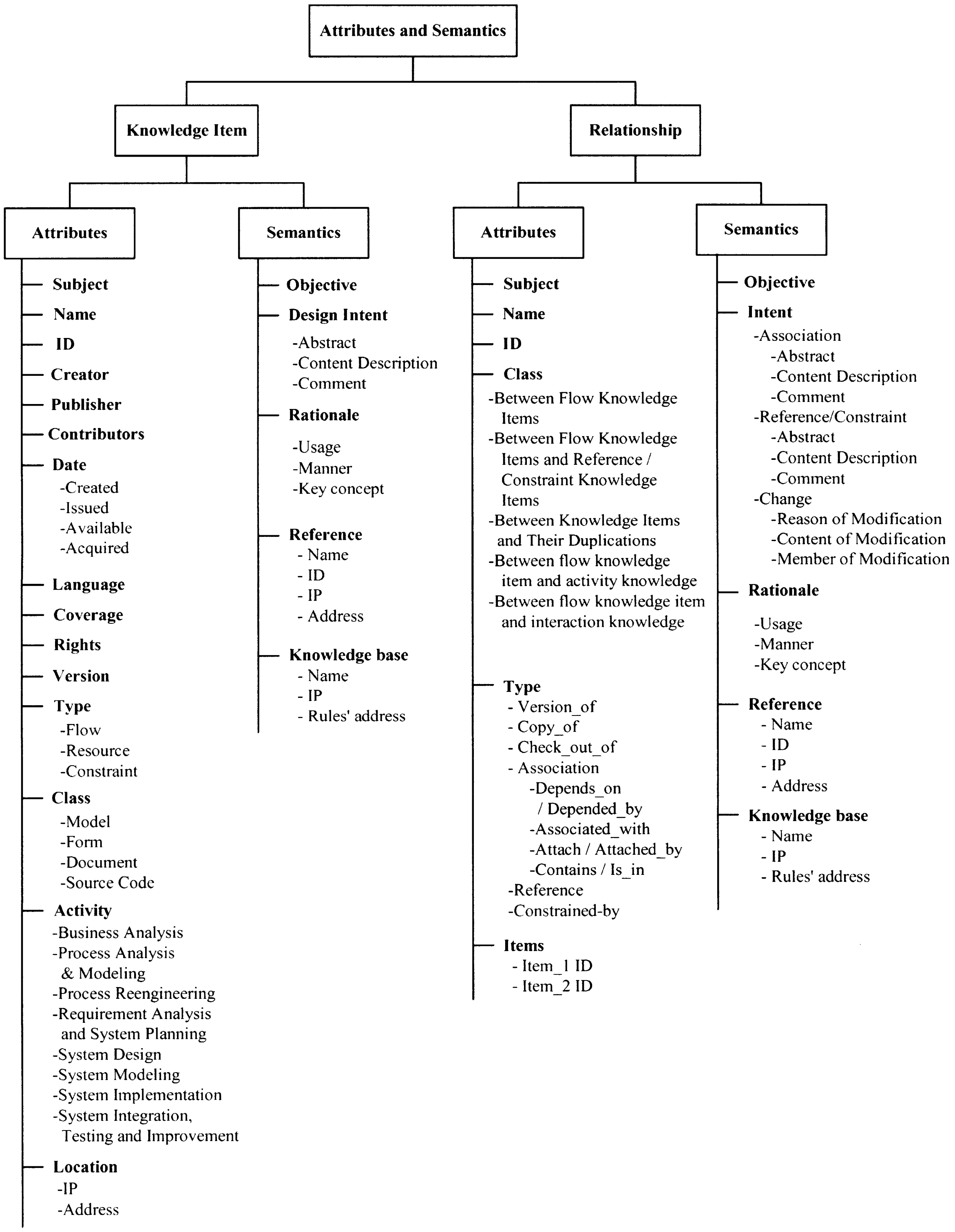


Fig. 7. Attributes and semantics.

associated_with, attached to/attached_by, and contains/is_in, while the second type of relationships include refer_to, and constrained_by. Examples of the third type of relationships include copy_of, version_of, and check_out_of. Both the fourth type and fifth type relationships include depended_by and modified_by, respectively.

### *3.2.2. Attributes and semantics analysis*

An attribute describes a characteristic or property of a knowledge item or a relationship. As illustrated in Fig. 7, the attributes of a knowledge item include: item subjects that describe the topic of the content of the project, including item, item name, item ID, item creator namely the entity primarily responsible for creating the item, item publisher namely the entity responsible for making the item available, item contributors defining an entity responsible for contributing to the item content, item dates defining date associated with important event in the life cycle of the item, including its creation, issue, available, and acquisition, item coverage defining the extent or scope of the item content, item rights defining information about rights held in and over the item, item version, item type differentiating the item in terms of nature or genre of flow, resource and constraint in an activity, item class differentiating the item in terms of class or category of the content of the model, form, document and source code, activity that creates the item, and the location where the item is located. Similarly, the attributes for a relationship include: relationship subject, name, ID, class, type, and items involved in the relationship.

Semantics indicate the meanings or intentions of knowledge items or relationships. As shown in Fig. 7, the semantics of a knowledge item include: objective, design intent, rationale, references, and knowledge bases. "Objective" defines the purpose, motivation or functions of a knowledge item; "Design intent" expresses the basic ideas behind the objective, including abstract, content description, and comment; "Rationale" indicates the rationales supporting ideas, including usage, manner, algorithm, and key concept; "References" list the reference materials used to create the knowledge item, and "Knowledge bases" contain the detailed principles, rules and explanations of the rationales. Similarly, relationship semantics include objective, intent, rationale, references, and knowledge bases. The difference in intent between a knowledge item and a relationship is that the intent can be further classified into association, reference/constraint and change. The content of both the association and reference/constraints includes abstract, content description, and comment, and the content of change includes reason for modification, content of modification, and member name.

## *3.3. Conceptual knowledge modeling*

### *3.3.1. Knowledge modeling*

Knowledge modeling aims to define the attributes and semantics details as well as underlying knowledge of knowledge items into levels of abstraction to facilitate knowledge storage, management and sharing. As shown in Fig. 8, a conceptual knowledge model is developed based on the concepts of knowledge abstraction and deepening.

Knowledge abstraction for a knowledge item is divided into three levels. The first level is the *attributes level* and includes the attributes of a knowledge item, as discussed in Section 3.2.2, which can facilitate knowledge management and knowledge searching. The second level is the *semantics level* and includes the semantics of

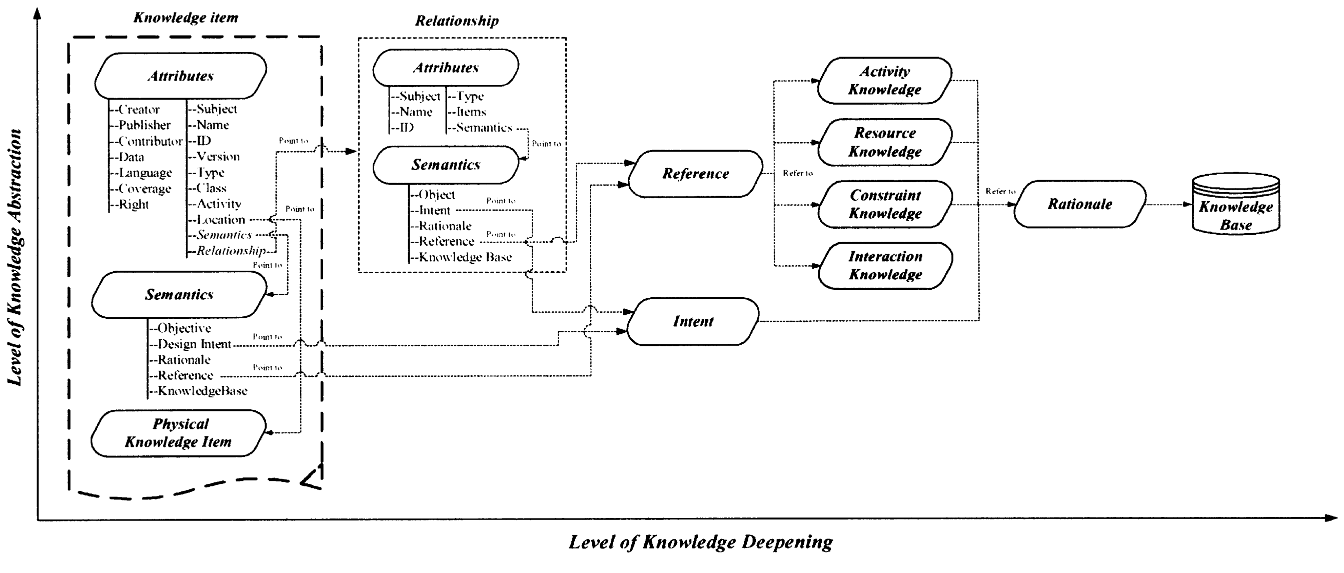


Fig. 8. Conceptual knowledge model.

a knowledge item, which offer details of a knowledge item to knowledge workers and thus help them to understand the knowledge item. Finally, the third level is the *physical item level* and includes physical knowledge items, such as the process model, system functional model, and software components. Additionally, knowledge abstraction for a relationship is divided into two levels. The first level is the *attributes level*, which includes the attributes of a relationship, as discussed in the previous sub-section, which can facilitate the tracking of related knowledge items. Meanwhile, the second level is the *semantics level*, which includes the semantics of a relationship, to provide relationship details to knowledge workers and thus help them to understand the relationship between knowledge items.

On the other hand, knowledge deepening divides roughly into three levels. The first level includes a knowledge item and a relationship, while the second level contains underlying knowledge, including their references and intents. Both the first and second levels point to design intent and reference in semantics. Within the second level, references can be further classified into four types, namely activity knowledge, resource knowledge, constraint knowledge, and interaction knowledge. However, each type of reference knowledge and intent knowledge can be derived from rationale provided by the knowledge base.

### 3.3.2. *Tag-based knowledge representation scheme*

In this research, a "tag-based knowledge representation scheme" is proposed to realize the above-mentioned conceptual knowledge model. A "Tag" is viewed as an attachment to a knowledge item or a relationship, which indicates its relevant knowledge details. Fig. 9 illustrates the scheme of knowledge tags. A "Tag" comprises a tag name, a tag ID, a declarative slot, and a procedural frame. The declarative slot contains the attributes of a knowledge item or a relationship, such as subject, name, ID, and so on. The attributes are represented in terms of entity relationship model (E-R Model). Meanwhile, the procedural frame on a tag includes the semantics of the knowledge item, namely objective, design intent, rationale, reference, and knowledge base. The textual format (HTML Format) is employed to represent the content of the objective, design intent, rationale, and reference, and the rule format is employed to represent the content of the knowledge base.

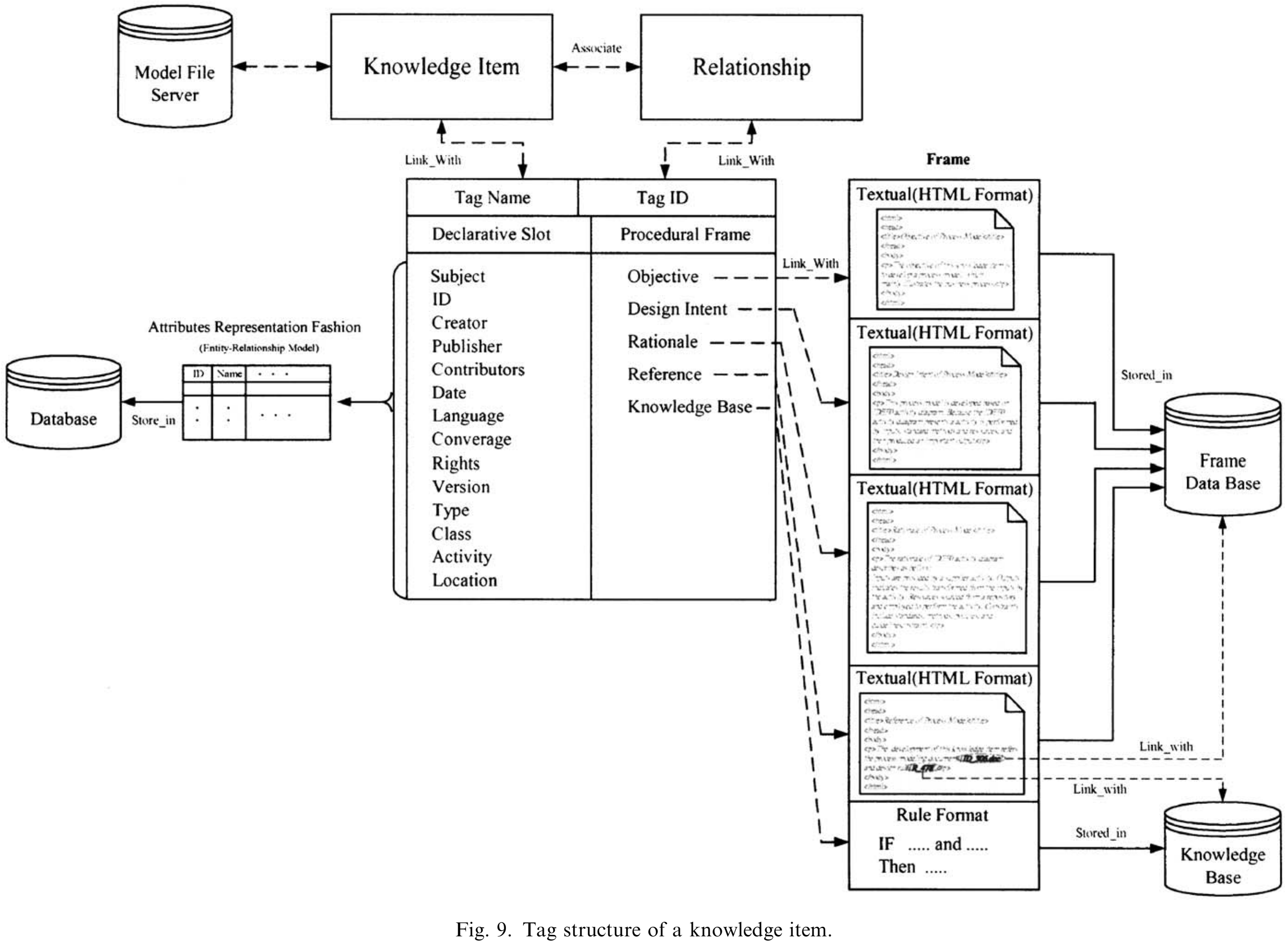


Fig. 9. Tag structure of a knowledge item.

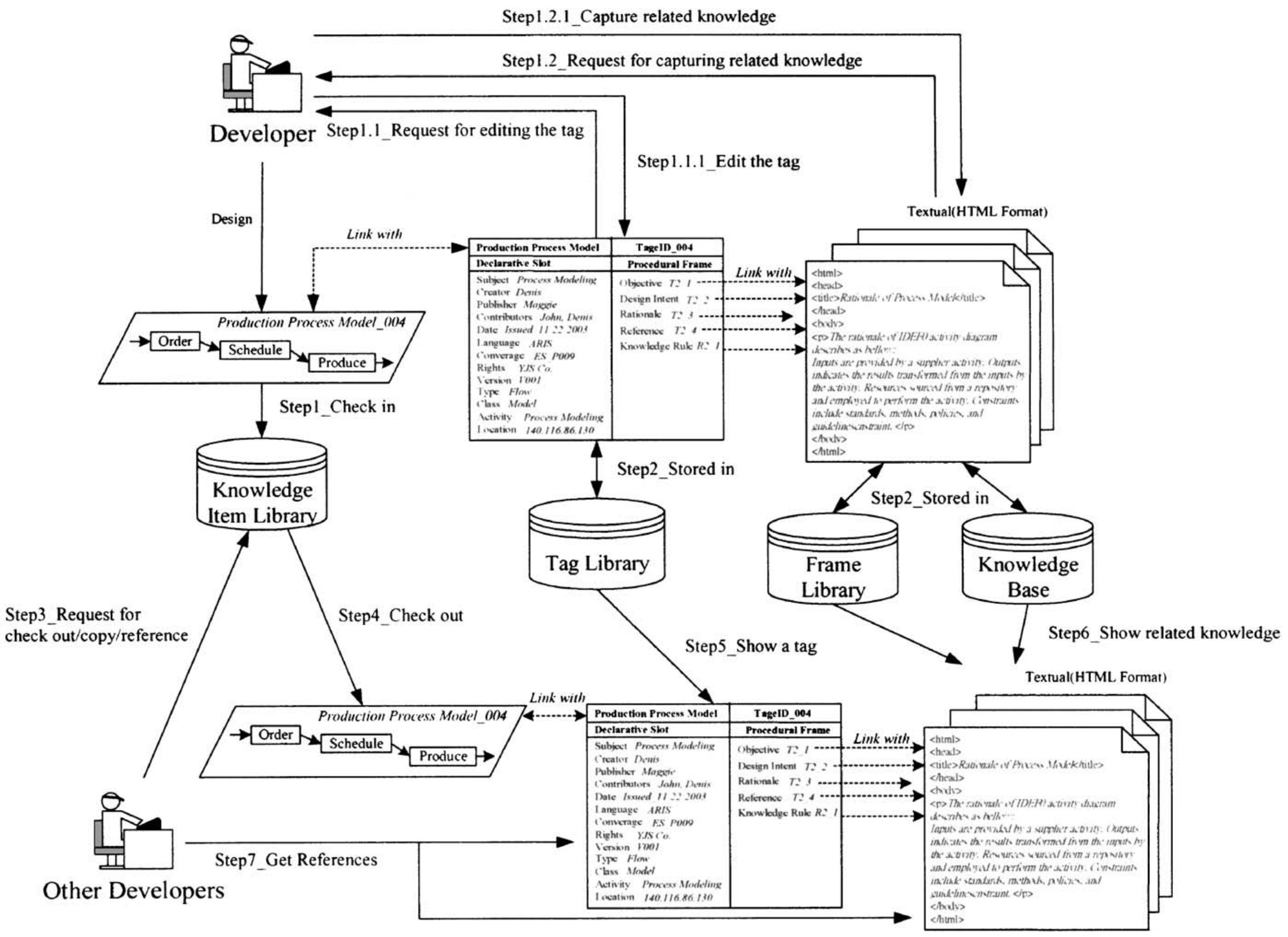


Fig. 10. The scenario of the knowledge model.

## 3.4. The scenario of the knowledge model

This section aims to describe the scenarios of how the knowledge model is to be used. As illustrated in Fig. 10, the scenario is initiated by checking the process model into the knowledge item library. At the same time, a tag and knowledge related to the knowledge item (i.e., the process model) are edited and captured by the developer.

On the other hand, the knowledge items in the knowledge item library can be checked out, copied, and referenced while retrieving a tag and related knowledge. The knowledge item and related knowledge are good references for other developers.

## 4. Development of the distributed knowledge model

This section first develops and analyzes the strategy and framework characterization, respectively, of knowledge management in the context of collaborative development and implementation of enterprise systems. The distributed knowledge management framework is then developed based on the strategy and framework characterization. According to the distributed knowledge management framework thus developed, a distributed knowledge model is designed using the conceptual modeling technique.

### 4.1. Knowledge management framework

#### 4.1.1. Strategy and characterization

Collaborative development process is an approach to integrating resources from different enterprises to overcome the difficulties faced by conventional enterprise system development approaches. To support the management and sharing of knowledge in a collaborative development and implementation process, a distributed knowledge management strategy is proposed. It includes characteristics as follows:

- *Project-based*: Partners in a collaborative development and implementation process are working together within a limited project in a commonly agreed time frame with a common goal and well-defined responsibilities. The knowledge management activities in the process can be seen as a sub-project of the system development project. It is similarly project-based.

- *Flexible configuration*: The environment for knowledge management to support a collaborative development and implementation project can be quickly and easily reconfigured based on the changing of partners.
- *Hierarchical and recursive structure*: The activity of a collaborative development and implementation process can be itself a lower-order process and can be further decomposed into lower-order activities. This forms the hierarchical structure of a collaborative development and implementation process. To meet the needs of the hierarchical development, the structure of knowledge management is necessarily configured in a hierarchical and recursive fashion.
- *Distributed and cooperativeness*: Decision authority and responsibility for knowledge management and sharing are distributed among levels of collaborative development and implementation projects. However, the knowledge management of individual project is conducted in collaboration with that of other projects to achieve higher-level goals through communication and coordination.

#### 4.1.2. Distributed knowledge management framework

In this section, a distributed knowledge management framework is proposed to reflect the flexibility, distributed and cooperativeness, and dynamic-configurability of collaborative development and implementation of enterprise systems.

First, the structure of collaborative development and implementation is identified to facilitate the design of knowledge management framework. Since the collaborative development and implementation is managed on a project basis, a collaborative development and implementation project may consist of one or more processes, each of which is in turn formed by one or more activities. An activity can be a primitive activity or a team activity. A primitive activity is performed by individuals, while a team activity is conducted by a team. According to the above findings, a collaborative development and implementation project can be defined in terms of a hierarchical structure, which reflects the characteristics of collaboration.

To support levels of knowledge management in collaborative development and implementation of enterprise systems, a *team knowledge agent,* a *personal knowledge agent,* and a *repository management agent* are designed. The *team knowledge agent* may perform knowledge management at levels of a project, a process, or a team activity, as well as conducts coordination with other agents. A team knowledge agent also owns functions for project, process, or team-wide knowledge management system configuration, knowledge item definition, and security control. By connecting lower-order knowledge management agents, a team knowledge agent can monitor and support the knowledge management on its subordinate. A *personal knowledge agent* is responsible for managing knowledge items of an activity for each project team member, as well as behaves as the interface between team members and team knowledge agent. A repository management agent is responsible for managing the knowledge entities defined in the knowledge model as discussed in Section 3, and maintaining their relationships.

Two levels of knowledge repositories are designed for levels of knowledge storage. They are team repositories and personal repositories. According to the knowledge model developed in Section 3, a team repository contains a tag library, a knowledge item library, a relationship library, a frame library, a knowledge base and a database. A *personal repository* includes *a workbench* and one or more *bins.* A workbench is the space where an individual team member performs a task as part of development activity. Bins are private storage areas for individual members of a project team. The team member can organize and control his or her work by taking things off a workbench and storing them in a bin.

To facilitate knowledge navigation, a knowledge map is designed as the global directory for knowledge items in a collaborative development and implementation project. The knowledge map is defined by a project administrator and is shared and maintained by team knowledge agents as the project is being conducted.

Fig. 11 shows the structure of the distributed knowledge management framework.

### 4.2. Distributed knowledge model

According to the tag-based knowledge representation scheme and the distributed knowledge management framework discussed in Section 3, a distributed knowledge model is designed as shown in Fig. 12.

A knowledge map is indeed a meta-model that shows the global view of the organization of team repositories and distribution of the knowledge items in a development project. The team knowledge repository is a directory that illustrates the personal knowledge repositories and the category of knowledge items and their knowledge details of a process or a team activity. A personal knowledge repository is also viewed as a directory or a meta-model indicating the category of knowledge items and their knowledge details of an activity.

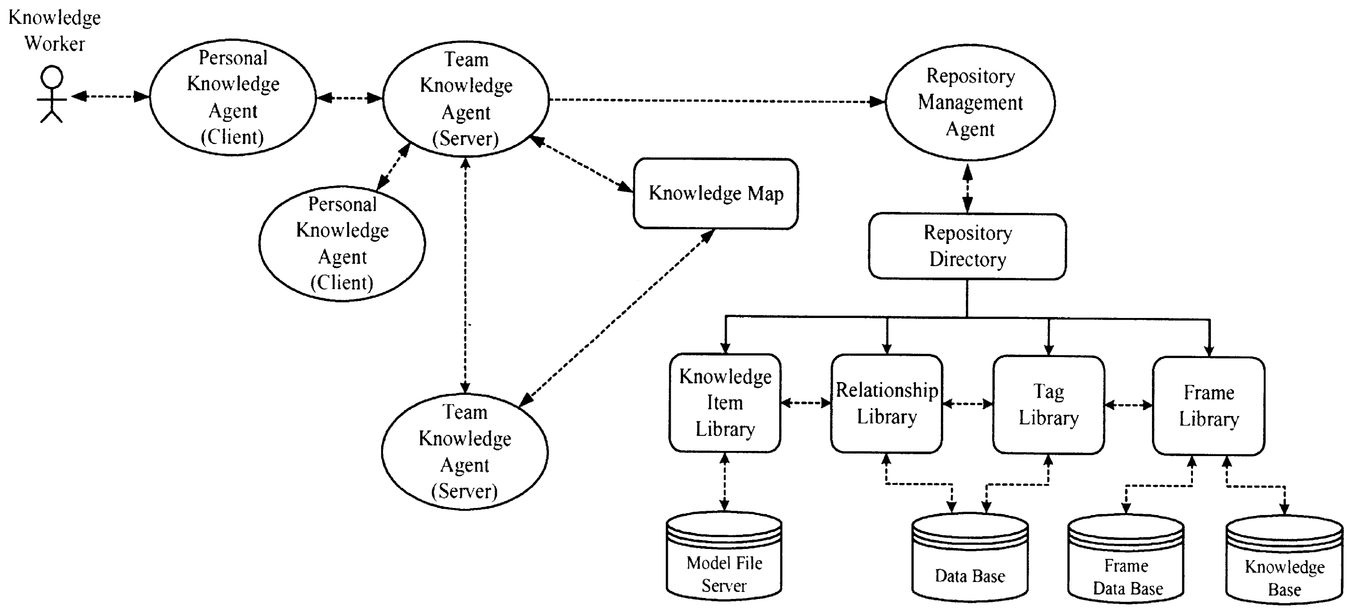


Fig. 11. Distributed knowledge management framework.

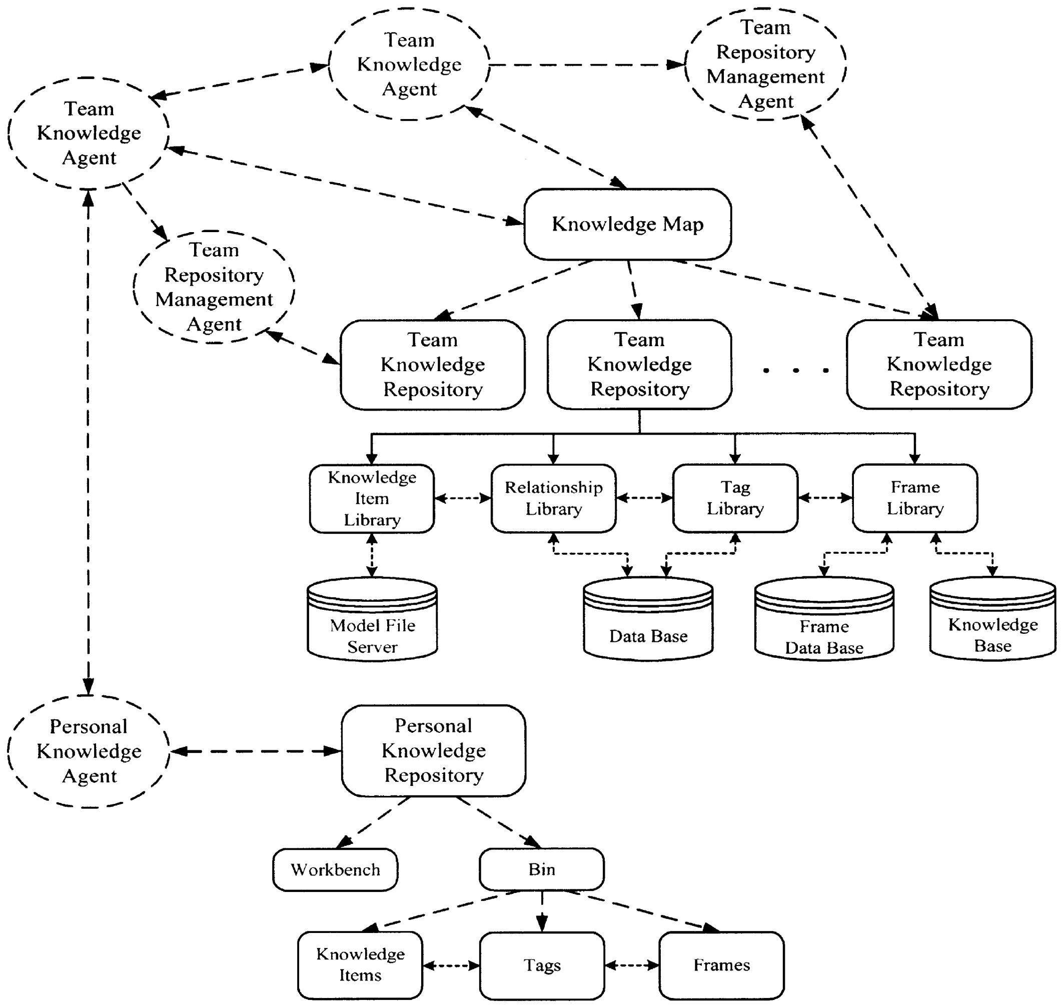


Fig. 12. Distributed knowledge model.

## 5. Distributed knowledge modeling

Based on the conceptual distributed knowledge model discussed in Section 4, this section presents the modeling of the proposed distributed knowledge model from a system development perspective using object-oriented modeling techniques. According to Rumbaugh's object modeling technique (OMT) [20–22], the proposed distributed knowledge model is developed in terms of object model, dynamic model, and object model design.

### 5.1. Object modeling

The purpose of object modeling is to represent the elements involved in distributed knowledge management and their relationships. An object model defines the static, structural, and data aspects of the proposed model in terms of objects and relationships, and is represented by an object diagram that consists of object classes, links and associations. An object class describes a group of objects that have common attributes, operations and semantics. A link is a physical or conceptual connection between object instances. An association describes a group of links that have a common structure and semantics. Two of the most commonly used associations are generalization and aggregation. Aggregation is the "part_of" relationship in which lower-level objects are associated as a higher-level aggregate object. Generalization is the "is_a" relationship in which lower-level subclasses are specialization of their super-class.

In Rumbaugh's notation, an object-class is indicated by a rectangular box divided into three regions: class names, list of attributes, and list of operations. Meanwhile, an object is denoted by a rectangular box with rounded corners. Additionally, associations

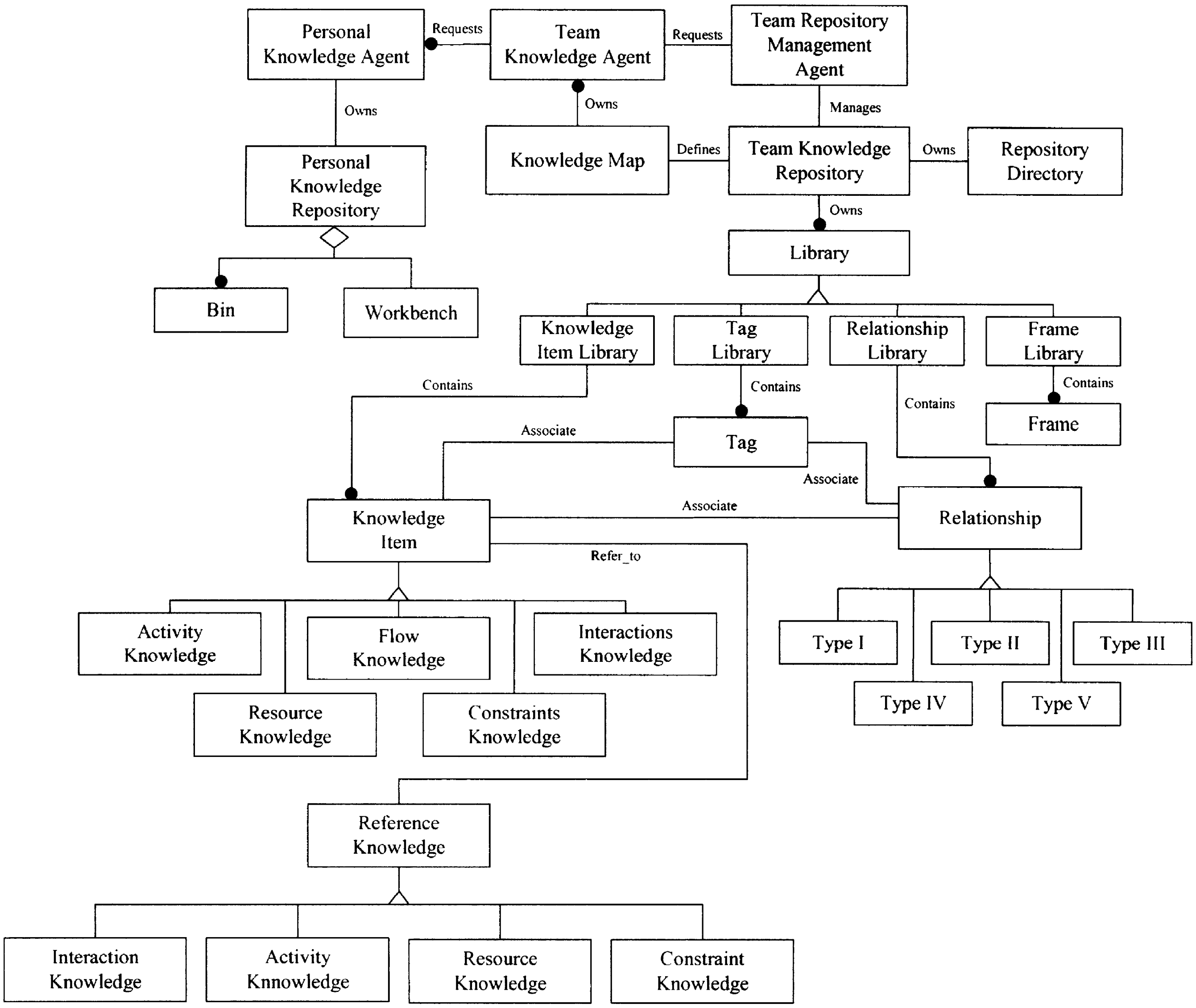


Fig. 13. Object model.

between classes are indicated with a line, which is associated with a verb in a problem statement. Similarly, the OMT notation for a link is a line between objects. Aggregation is represented like association, expect that a small diamond indicates an assembly end of the relationships. Finally, the generalization is indicated by a triangle connecting a super-class to its sub-classes.

According to the above-mentioned concepts of object model, an object distributed knowledge model is transformed from the conceptual knowledge model using object-oriented techniques as illustrated in Fig. 13. The distributed knowledge model is associated with a series of team knowledge agents, personal knowledge agents, team repository management agents, team knowledge repositories, personal knowledge repositories, and a global knowledge map. A team knowledge repository has many libraries, which can be classified as the knowledge item library, the relationship library, the tag library, and the frame library. A personal knowledge repository, consisting of a workbench and more bins, are the logical spaces that belong to an activity.

Meanwhile, the knowledge items can be further classified into the classes of activity knowledge item, resource knowledge item, flow knowledge item, constraint knowledge item, and interaction knowledge item. Furthermore, the knowledge item class can refer to the classes of interaction knowledge, activity knowledge, resource knowledge, and constraints knowledge. On the other hand, the relationship class is further classified into five subclasses. They are (i) relationships between flow knowledge items, i.e., depends_on, associated_with, attach_to etc.; (ii) relationships between flow knowledge items and reference knowledge items/constraint knowledge items, i.e., refer_to; (iii) relationships between knowledge items and their duplications, i.e., version_of, copy_of and check_out_of etc.; (iv) relationships between flow knowledge item and activity knowledge, i.e., depended_by, and (v) relationships between flow knowledge items and interaction knowledge, i.e., modified_by.

### 5.2. Dynamic modeling

The previous section examined the static structure of the proposed distributed knowledge model by identifying the structure and interrelationships of the objects in it. This section employs the dynamic modeling techniques of OMT to describe changes to the objects and their relationships over time. The dynamic modeling mainly includes event identification and state transition diagram development.

#### 5.2.1. Events and states

The object model presented above described the possible patterns of objects, attributes, and links that exist in the proposed distributed knowledge management framework. The attribute values and links of an object at a given moment are called its "state". Over time, knowledge workers may perform operations that change these states. On the other hand, "events" can be defined according to individual stimulus from outside the system, and are used to influence objects. The following are portions of the events from the knowledge worker to the distributed knowledge model:

- *External events related to distributed knowledge management*:
  Check in a knowledge item to library
  Check out a knowledge item from library

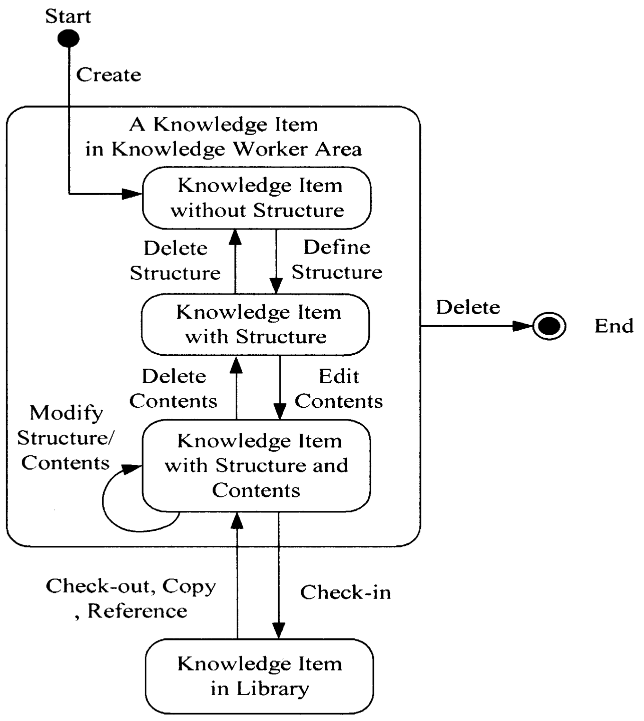

Start
Create
A Knowledge Item in Knowledge Worker Area
Knowledge Item without Structure
Delete Structure
Define Structure
Knowledge Item with Structure
Delete
End
Delete Contents
Edit Contents
Modify Structure/ Contents
Knowledge Item with Structure and Contents
Check-out, Copy , Reference
Check-in
Knowledge Item in Library

Fig. 14. State diagram of a knowledge item.

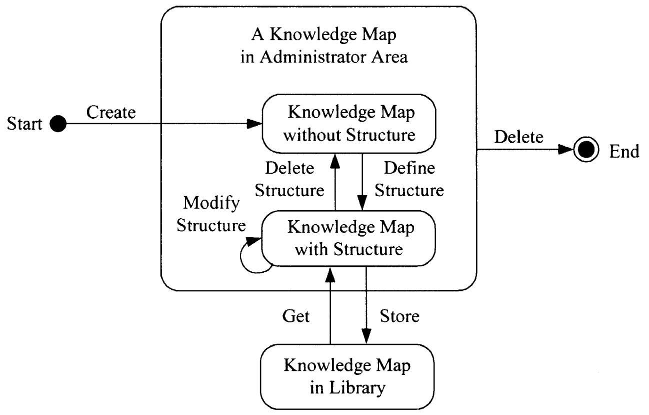


Fig. 15. State diagram of the knowledge map.

Reference a knowledge item from library
Copy a knowledge item from library
Put a knowledge item into a bin
Get a knowledge item from a bin
Create/Define/Delete project
Create/Define/Delete project team
Create/Define/Delete team knowledge library
Create/Define/Edit/Modify/Delete knowledge items

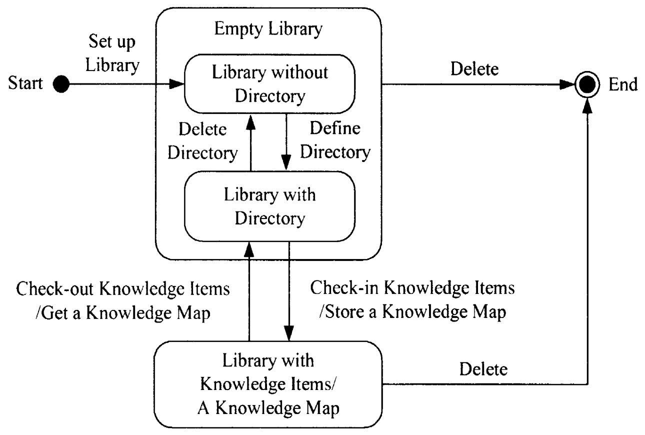

Start
Set up Library
Empty Library
Library without Directory
Delete Directory
Define Directory
Library with Directory
Delete
End
Check-out Knowledge Items /Get a Knowledge Map
Check-in Knowledge Items /Store a Knowledge Map
Library with Knowledge Items/ A Knowledge Map
Delete

Fig. 16. State diagram of the library.

Create/Define/Delete/Modify the knowledge map
Set up/Delete personal work area (knowledge worker area)

- *Internal events related to distributed knowledge management*:
  Verify user's password
  Check a knowledge item status
  Check user (knowledge workers/knowledge administrator) authority
  Look up the knowledge map
  Look up library directory
  Access/show the tag, attributes, semantics, and physical model of a knowledge item
  Notify change

### 5.2.2. State diagram

The events identified in the above are then used to define the state diagram of each object class. A state diagram that links states through events describes the behavior of a single class of objects. Using the notations of Rumbaugh's methodology, this study presents a state diagram as a graph in which the nodes are states and the directed arcs are transitions labeled by event names. Notably, states are represented in terms of a rounded

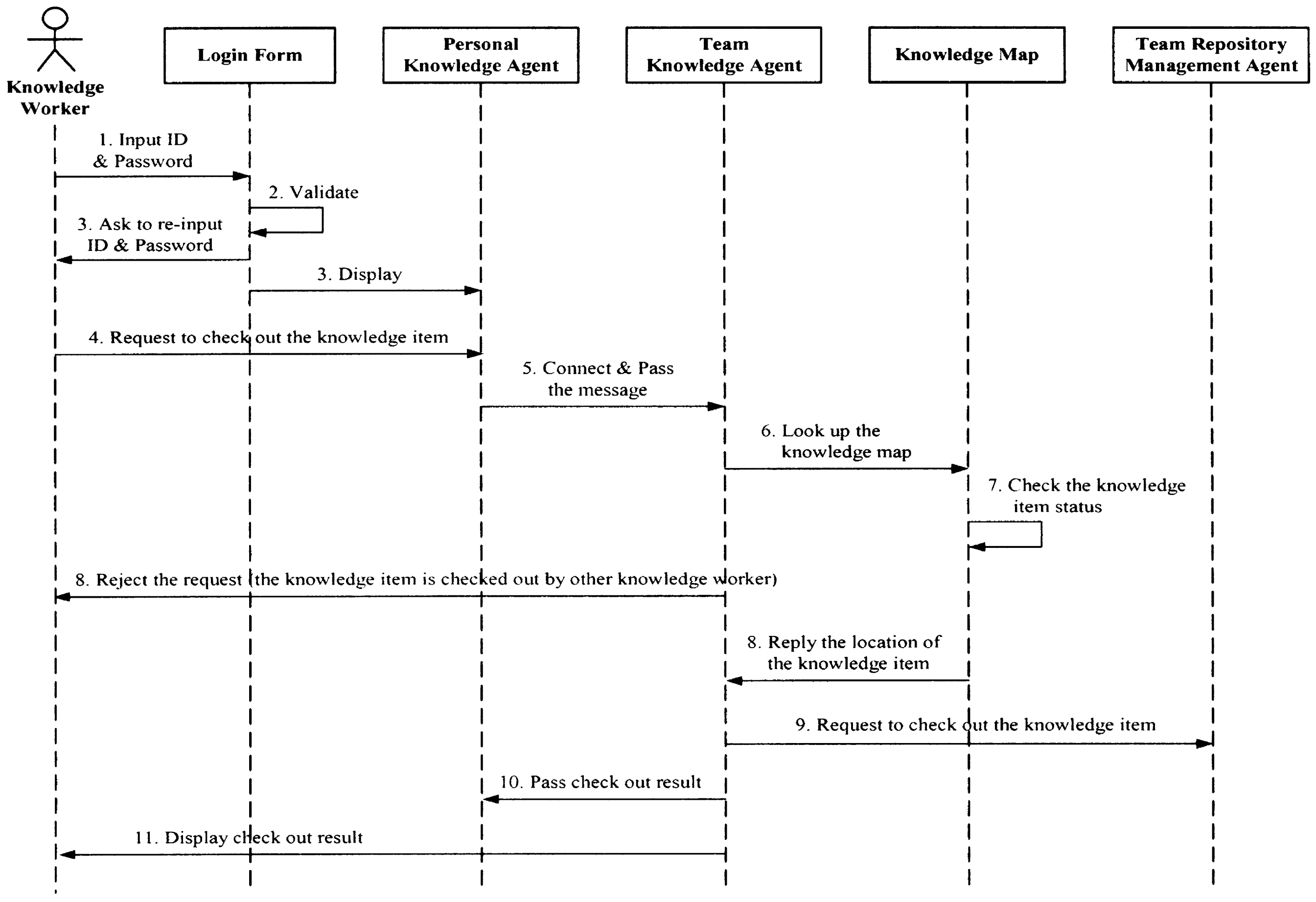

Knowledge Worker
Login Form
Personal Knowledge Agent
Team Knowledge Agent
Knowledge Map
Team Repository Management Agent
1. Input ID & Password
2. Validate
3. Ask to re-input ID & Password
3. Display
4. Request to check out the knowledge item
5. Connect & Pass the message
6. Look up the knowledge map
7. Check the knowledge item status
8. Reject the request (the knowledge item is checked out by other knowledge worker)
8. Reply the location of the knowledge item
9. Request to check out the knowledge item
10. Pass check out result
11. Display check out result

Fig. 17. Interactive sequence diagram of "Check out".

box containing an optional name. Meanwhile, a transition is drawn as an arrow from the receiving state to the target state, with a label on the arrow indicating the event causing the transition. All transitions leaving a state must correspond to different events.

Fig. 14 shows the state diagram of a knowledge item, which can be in a knowledge worker area or a library. Generally, the creation of a knowledge item without a structure begins in a knowledge worker area, and its structure and contents are defined and edited by the knowledge worker in that area. Mature knowledge items are stored in a library. Furthermore, knowledge workers can obtain knowledge items from libraries to modify their structure or contents.

Fig. 15 illustrates the state diagram of the knowledge map. The creation of the knowledge map with no structure begins in the administrator area. The administrator then adds the structure to the knowledge map that has no structure. Subsequently, the knowledge map that has structures is stored in a library. Notably, the

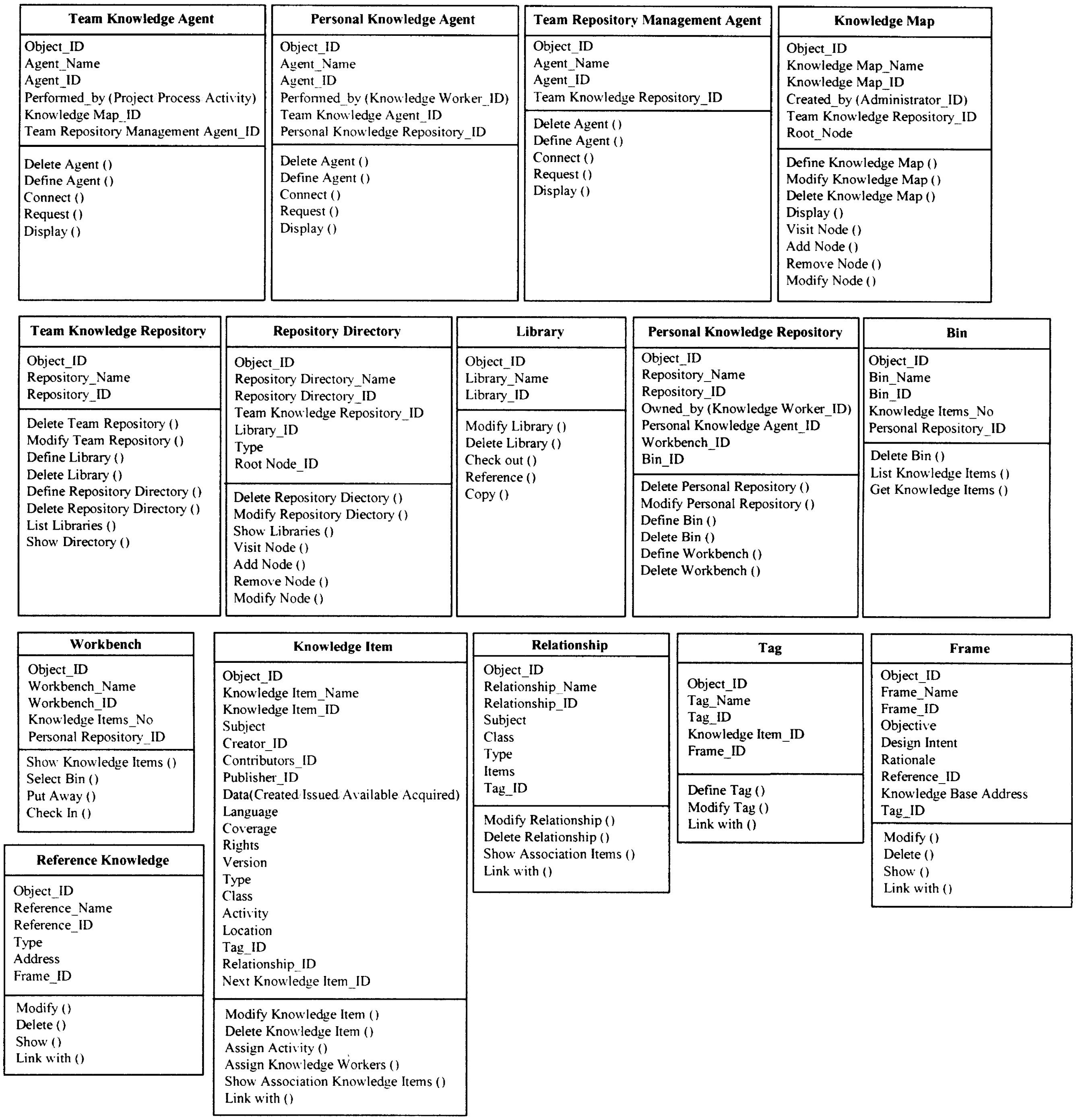


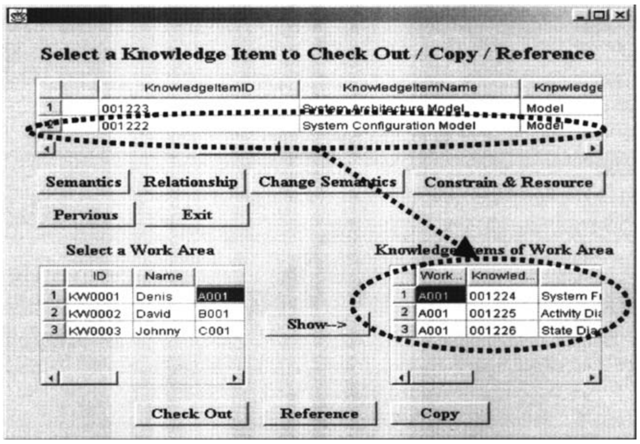
Fig. 18. Part of classes of the distributed knowledge model.

administrator is able to access the knowledge map to modify their structure.

Fig. 16 illustrates the state diagram of the library. An empty library without category is created when the project leader sets up a library for the distributed project team. Each of these libraries can be classified using several storage areas based on the distributed project processes and distributed project activities. Examples of directories include the knowledge map and knowledge items, and the knowledge item directory can further include the tag, relationship and frame.

### 5.2.3. *Interaction diagrams*

Interaction diagrams mainly describe the interactive relationships between objects. Interaction modeling can be used to clearly identify the operations of each object. As shown in Fig. 17, interaction modeling mainly illustrates the interactive sequence diagram of "Check out". The objects of interaction modeling include the login form, personal knowledge agent, team knowledge agent, knowledge map, and team repository management agent. The interactive relationships among these objects are displayed in Fig. 16.



Select a Knowledge Item to Check Out / Copy / Reference

| | | KnowledgeItemID | KnowledgeItemName | Knpwledge |
|---|---|---|---|---|
| 1 | | 001223 | System Architecture Model | Model |
| 2 | | 001222 | System Configuration Model | Model |

Select a Work Area

| | ID | Name | |
|---|---|---|---|
| 1 | KW0001 | Denis | A001 |
| 2 | KW0002 | David | B001 |
| 3 | KW0003 | Johnny | C001 |

Knowledge Items of Work Area

| | Work... | Knowled... | |
|---|---|---|---|
| 1 | A001 | 001224 | System Fr |
| 2 | A001 | 001225 | Activity Dia |
| 3 | A001 | 001226 | State Diag |

Fig. 19. User interface—"Before Check-Out".

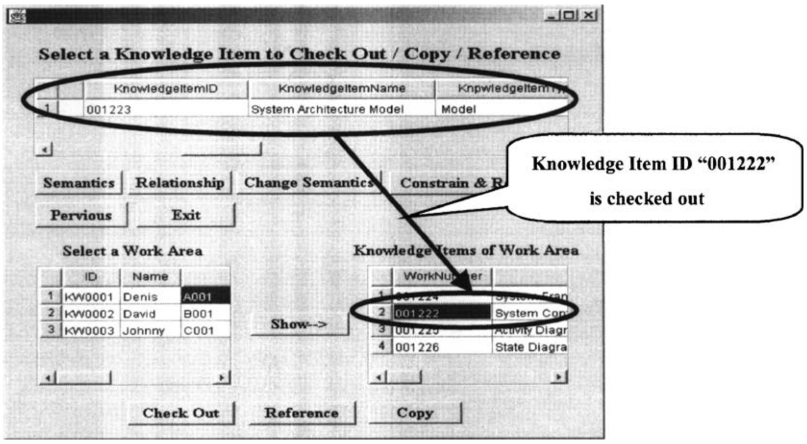
### 5.3. *Object model design*

Based on the object-oriented modeling, the object classes are designed as shown in Fig. 18. Each object class can be defined in terms of class name, class attributes and class operations/methods. Fig. 18 displays part of the classes. A directed graph and a tree are employed to represent the structure of the knowledge map and repository directory, respectively.

## 6. System implementation

Based on the proposed distributed knowledge model, a prototype knowledge management system for collaborative enterprise system development and implementation was implemented in a virtual concurrent engineering environment created by the Enterprise Engineering and Integration Research Lab at National Cheng Kung University, Taiwan, ROC, and the Computer Integrated Manufacturing Research Lab at National Kaohsiung University of Applied Sciences, Taiwan, ROC. The computer hardware used for the experiment was an Acer Altos 9000 PC™ Server and an Acer Power 590 h PC™ workstation networked with six PC clients under Windows-NT™. The software component employed for implementing this system was Java Builder 3.

Figs. 19 and 20 show two of the user interfaces of the knowledge management system for collaborative enterprise system development and implementation. Meanwhile, Fig. 19 shows the form used for "Before Check-out", while Fig. 20 shows the form used for "After Check-out".



Select a Knowledge Item to Check Out / Copy / Reference

| | | KnowledgeItemID | KnowledgeItemName | KnpwledgeItem |
|---|---|---|---|---|
| 1 | | 001223 | System Architecture Model | Model |

Select a Work Area

| | ID | Name | |
|---|---|---|---|
| 1 | KW0001 | Denis | A001 |
| 2 | KW0002 | David | B001 |
| 3 | KW0003 | Johnny | C001 |

Knowledge Items of Work Area

| | WorkNumber | |
|---|---|---|
| 1 | 001224 | System Fran |
| 2 | 001222 | System Con |
| 3 | 001225 | Activity Diagr |
| 4 | 001226 | State Diagra |

Fig. 20. User interface—"After Check-Out".

## 7. Conclusions and discussions

This study has presented the systematic steps to developing a distributed knowledge management model, including: (i) modeling and characterization of the process of developing and implementing a collaborative enterprise system, (ii) identification, analysis and representation of involved knowledge, and (iii) development of a distributed knowledge model for knowledge management in the collaborative development and implementation environment of an enterprise system.

The proposed object-oriented distributed knowledge model has characteristics modularity, expandability and flexibility, as well as (1) provides an effective way to record not only explicit knowledge but also tacit knowledge, (2) offers a convenient manner for knowledge reuse, and (3) can be applied to environment for other knowledge intensive works. For example, engineering design is really knowledge intensive. It includes the tasks of conceptual design, detailed design, engineering analysis, assembly design, process design, and performance evaluation. The execution of each task requires various aspects of knowledge as well as experience. Therefore, apply the developed knowledge model to engineering design so as to capture product information and related engineering knowledge from engineering designers is urgent in engineering knowledge management.

The results of this study will enable the development of a collaborative knowledge management system to support the collaborative development and implementation of an enterprise system, and consequently increase enterprise competitiveness.

## Acknowledgements

The authors would like to thank the National Science Council, Taiwan, ROC for financially supporting this research under Contract No: NSC 91-2212-E-006-064.

## References

[1] Allen P, Forst S. Computer-based development for enterprise systems. Cambridge, UK: Cambridge University Press; 1998.
[2] Jeff S. Enterprise application integration and complex adaptive systems. Commun ACM 2002;45:59–64.
[3] Michael JS. Building an e-business from enterprise systems. Inf Syst Front 2000;2:7–17.
[4] Thomas HD. The future of enterprise system-enabled organizations. Inf Syst Front 2000;2:163–80.
[5] Chen YM, Liang MW. Design and implementation of a collaborative engineering information system for allied concurrent engineering. Int J Comput Integrated Manuf 1999;13(1): 11–30.
[6] Davidow WH, Malone MS. The virtual corporation. USA: Harper Collins Publishers; 1992.
[7] Prasad B. Concurrent engineering fundamentals. Integrated product and process organization, vol. I. Englewood Cliffs, NJ: Prentice-Hall; 1996.
[8] Prasad B. Concurrent engineering fundamentals. Integrated product and process organization, vol. II. Englewood Cliffs, NJ: Prentice-Hall; 1997.
[9] Barthelme F, Ermine JL, Rosenthal-Sabroux C. An architecture for knowledge evolution in organisations. Eur J Oper Res 1998;109(2):414–27.
[10] Basu A. Perspectives on operations research in data and knowledge management. Eur J Oper Res 1998;111(1):1–14.
[11] Carayannis EG. The strategic management of technological learning in project/program management: the role of extranets, intranets and intelligent agents in knowledge generation, diffusion, and leveraging. Technovation 1998;18(11):697–703.
[12] Drew S. Building knowledge management into strategy. Making sense of a new perspective. Long Range Plann 1999;32:130–6.
[13] Purser RE, Pasmore WA. Organizing for learning. Research in organization or change and development. London: JAI Press Inc; 1992. p. 37–114.
[14] Studer R, Benjamins VR, Fensel D. Knowledge engineering: principles and methods. Data Knowledge Eng 1998;25(1–2): 161–97.
[15] Chen YM, Hsiao YT. A collaborative data management framework for concurrent product and process development. Int J Comput Integrated Manuf 1997;10(6):446–69.
[16] Chen YM, Liao CC, Prasad B. A systematic approach of virtual enterprising through knowledge management techniques. J Concurrent Eng Res Appl 1998;6(3):225–44.
[17] Chen YM, Lee RS, Ho CT. A framework for integration of design and knowledge based environment. J Chin Soc Mech Eng 1996;17(6):587–99.
[18] Chen YM, Jan YD. Enabling allied concurrent engineering through distributed engineering information management. Robotics Comput Integrated Manuf 2000;16:9–27.
[19] Chen YM, Tsao TH. A structured methodology for implementing engineering data management. Robotics Comput Integrated Manuf 1998;14:275–96.
[20] Booch G, Rumbaugh J, Jacobson I. The unified modeling language user guide. Reading, MA: Addison-Wesley; 1999.
[21] Jacobson I, Christerson M, Jonsson PM, Overgaard G. Object oriented software engineering: a use case driven approach. Reading, MA: Addison-Wesley; 1992.
[22] Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. Object-oriented modeling and design. Englewood Cliffs, NJ: Prentice-Hall; 1992.