Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 79 (2006) 1469–1482
The Journal of Systems and Software
www.elsevier.com/locate/jss

Semantic component networking: Toward the synergy of static reuse and dynamic clustering of resources in the knowledge grid ☆

Hai Zhuge *

Hunan Knowledge Grid Lab, Hunan University of Science and Technology, Hunan, China
China Knowledge Grid Research Group, Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704-28, Beijing 100080, China

Received 5 January 2005; received in revised form 12 February 2006; accepted 19 February 2006
Available online 5 May 2006

Abstract

Model is a kind of codified knowledge that has been verified in solving problems. Solving a complex problem usually needs a set of models. Using components, the composition of a set of closely related models, could enhance the efficiency of solving problems as components are experience in form of knowledge network. Mappings between components vary with the relevancy of problems. Such mappings can help retrieve appropriate components when solving new problems. This paper investigates the relationship between components to identify these mappings by a set of semantic links and develops relevant rules to form a mechanism for semantically networking and using components. Flexible component reuse can be realized by semantic retrieval and rule reasoning. Extending the notion of component to a general service mechanism, this paper further investigates the organization, reuse and clustering of components in form of Web services and research spaces. The semantic component networking overlay supports intelligent e-Science Knowledge Grid applications by the synergy of static reuse and dynamic clustering of various research spaces and resources on-demand.

Keywords: Component; Model; Reasoning; Reuse; Rules; Knowledge grid; Service; Web

1. Introduction

Different from tacit knowledge, explicit knowledge can be codified for reuse and sharing. Concept, axiom, rule, method, and theory are codified knowledge. Model is a kind of codified knowledge that has been verified in solving problems.

Solving a complex problem in some areas (e.g., making a large-scale military plan) usually needs the synergy of a set of models. People have the tendency to use existing solutions or experience rather than to find appropriate models and then to compose them during problem-solving. A component is a type of problem-solving experience—a network of codified knowledge for solving a complex problem.

The component-based problem-solving has three main advantages: (1) Problem-oriented. People can concentrate on problem analysis and making problem abstraction so as to hold the key to the problem. Otherwise, people have to focus on local problems and deal with technical details. (2) Efficient. It is more efficient to use components rather than to retrieve solutions and to compose them when making decisions. (3) Experience-based. A component encapsulates problem-solving experience, which can be used when solving new problems.

This paper investigates a general component methodology from the following aspects:

(1) The semantic relationships between components and relevant logical mechanism for flexible component reuse.

☆ Research work was supported by the National Basic Research Program of China (973 project no. 2003CB317000) and the National Science Foundation of China.
* Tel.: +86 1062565533; fax: +86 1062567724.
E-mail address: zhuge@ict.ac.cn

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.02.033

(2) A component overlay that supports flexible component reuse.
(3) The effectiveness and efficiency of the component overlay in supporting problem-solving.
(4) The synergy of the static reuse and dynamic clustering of components to form a scalable high-level semantic overlay for the Knowledge Grid.

Previous model base structures include the OO (Object-Oriented) models (Lenard, 1993; Novak, 1997; Rumbaugh et al., 1991), the frame structure model (Dolk and Konsynski, 1984), the network and relational models (Blanning, 1993), the knowledge-based structure model (Yau and Tsai, 1987), and the modelling language representation model (Hong et al., 1993). The OO models support a class-based inheritance, which facilitates the incremental definition of models and reuse of models. But, the implementation of the inheritance mechanism depends on the implementation language used (i.e., OOPL, Object-Oriented Programming Language). Unfortunately, many existing mathematical models are 'flat' and 'standard' (e.g., FORTRAN-based MSL/Math) due to domain-specific features. These models are difficult to be reengineered by using OOPLs or to be implemented as the OO paradigm by using non-OOPLs. The CORBA (Common Object Request Broker Architecture, http://www.omg.org/corba) and COM/DCOM (Common Object Model/Distributed Common Object Model, http://www.microsoft.com/com/) are system-level and language-irrelevant component interface standards, but they do not have the mechanism for reuse. Traditional model bases provide users or applications with exact retrieval mechanism. Efforts have been made to relax the exact retrieval, for example, the flexible model retrieval (Zhuge, 1998) and the flexible Web service (distributed deployment and execution of models) retrieval (Zhuge and Liu, 2004).

Experts are good at finding and using not only the relationships between problems but also the relationships between past experiences during problem-solving. To achieve this effect in problem-solving support systems, this paper proposes a set of basic semantic links between components to specify different extents and different views that may apply to when one component uses another component. A new component can be created by using an existing component from different views and to different extents. Based on these semantic links, a set of reasoning rules and a set of operation rules are developed for component reuse. A rule reasoning mechanism is developed to search for a proper component in a candidate set. To increase the flexibility of component reuse, an inexact factor (similarity degree) can be incorporated into the semantic link, and propose the corresponding rule reasoning formalism. The semantic links are used to construct multiple semantic overlays, which are language irrelevant and can support advanced applications.

2. Semantic links

2.1. Definition

A component can be specified by: a name (identifier), a signature (interface), and its implementation. The name reflects its behaviour category (Boisvert et al., 1985). The signature represents a mapping from the input type into the output type of the component. The implementation of a component is a network $C = \langle N, E, R\rangle$, where N is a set of nodes (the building blocks of the component), E is a set of edges (the relationship between nodes), R is a set of restrictions on N and E. A view of the component C, denoted as $View(C)$, is a sub-graph of the implementation network.

An edge can be regarded as a flow between two nodes. It includes the following six types: (1) sequential dependence (SD), i.e., one node is executed after another; (2) conditional sequential dependence (CSD), i.e., SD only occurs under a condition; (3) data dependence (DD), i.e., the input of one node depends on the output of the other; (4) conditional data dependence (CDD), i.e., DD only occurs under a condition; (5) function dependence (FD), i.e., a node is implemented by calling the functions of the other nodes; and, (6) conditional function dependence (CFD), i.e., FD occurs under a condition. We use $T(e)$ to denote the type of edge e, $T(e) \in \{SD, CSD, DD, CDD, FD, CFD\}$. Generally, we use '$n_1 \cdots T(e) \cdots \rightarrow n_2$' to denote '$n_1$ depends on n_2 with type T(e)'. A node can have an input restriction, which is either the 'AND' or the 'OR' of its input flows. A node can also have an output restriction, which is either the 'AND' or the 'OR' of its output flows.

Let $C' \leftarrow\alpha\rightarrow C$ be an isomorphism from component $C' = \langle N', E', R'\rangle$ into $C = \langle N, E, R\rangle$, α can be a mapping or a set of transformation function $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, and for any $n \in N$ and $n' \in N'$, there exists an $\alpha_i \in \alpha$ such that $\alpha_i(n') = n$. For an onto mapping from $C = \langle N, E, R\rangle$ into $C' = \langle N', E', R'\rangle$, $C \text{—}\beta\rightarrow C'$, $\beta = (\beta_1, \beta_2, \ldots, \beta_k)$, and for any $view(C') \in N'$, there exists a $view$(C) such that $\beta(view(C)) = view(C')$. A view can be a node or a network.

Definition 1. A semantic link between components C (ancestor) and C' (descendent) is defined by: (1) if $C' \leftarrow\alpha\rightarrow C$, then we say that C' is isomorphism with C, denoted as $C' \text{—}I\alpha\rightarrow C$; (2) if there exists a $View(C')$, such that $View(C') \leftarrow\alpha\rightarrow C$, then we say that C' totally contains C with α, denoted as $C' \text{—}T\alpha\rightarrow C$; (3) if there exists a $View(C)$ such that $C' \leftarrow\alpha\rightarrow View(C)$, then we say that C' partially contains C with α, denoted as $C' \text{—}P\alpha\rightarrow C$; (4) if there exist $View(C)$ and $View(C')$, such that $View(C') \leftarrow\alpha\rightarrow View(C)$, then we say that C' reduces together with C under α, denoted as $C' \text{—}R\alpha\rightarrow C$; and, (5) if there does not exist a $View(C)$ and a $View(C')$, such that $View(C') \leftarrow\alpha\rightarrow View(C)$, then we say that C' is irrelevant to C under α, denoted as $C' \text{—}\times\rightarrow C$.

Except empty, the other semantic factors (mapping or transformation) can be attached to a condition of behaviour compatibility, i.e., whether a descendant is behaviour

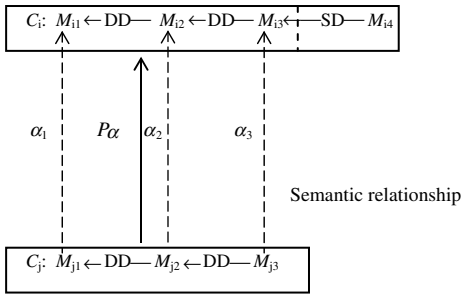

Fig. 1. A semantic link between two components. *Legend*: α_k: semantic link between M_{ik} and M_{jk} ($k = 1, 2, 3$), $\alpha = (\alpha_1, \alpha_2, \alpha_3)$, $P\alpha$: partial semantic link between C_i and C_j.

compatible with the ancestor or not (Zhuge, 1998). We herein add a letter 'B' in front of these factors to denote the condition. Then, the factor set of the semantic link is extended as: $\{BI\alpha, I\alpha, BT\alpha, T\alpha, BP\alpha, P\alpha, BR\alpha, R\alpha, \times\}$. A semantic link example is shown in Fig. 1, where C_i is implemented by composing four models (M_{i1}, M_{i2}, M_{i3}, and M_{i4}) with the data dependence and sequential dependence relationship, and C_j is implemented by composing three models (M_{j1}, M_{j2}, and M_{j3}) with data dependence relationship. If $M_{im} \text{—}\alpha_i\text{→} M_{jn}$ holds for $m, n = 1, 2, 3$, and then we have: $C_i \text{—}P\alpha\text{→} C_j$, $\alpha = (\alpha_1, \alpha_2, \alpha_3)$.

The semantic link is defined in terms of how (totally or partially) the descendant takes after the ancestor, and whether new nodes or edges have been added to the descendant or not. It can reflect a kind of relationship between any two components from different views. For example, a component C' takes after another component C with two different mappings: α and α', i.e., $C' \text{—}P\alpha\text{→} C$ and $C' \text{—}P\alpha'\text{→} C$ can be both hold.

2.2. The order of semantic links

Let '$\leqslant$' be the order on the extended semantic factor set (e.g., $R\alpha \leqslant BR\alpha$ means that $BR\alpha$ is stronger than $R\alpha$), we have the following five orders:

(1) $\times \leqslant R\alpha \leqslant BR\alpha \leqslant BP\alpha \leqslant BI\alpha$;
(2) $BR\alpha \leqslant BT\alpha \leqslant BI\alpha$;
(3) $R\alpha \leqslant P\alpha \leqslant I\alpha \leqslant BI\alpha$;
(4) $T\alpha \leqslant BT\alpha$; and,
(5) $R\alpha \leqslant T\alpha \leqslant I\alpha$.

The order relationship on the semantic factor set is graphically shown in Fig. 2, where the upper node is stronger than the related lower node. The graph shows that, for any two elements of the set, there exist a least-upper-bound (LUB) and a greatest-lower-bound (GLB). Hence, the relationship '$\leqslant$' constitutes a lattice on the semantic factor set.

Lemma 1. *The order relationship "$\leqslant$" on the set of the semantic links is a lattice.*

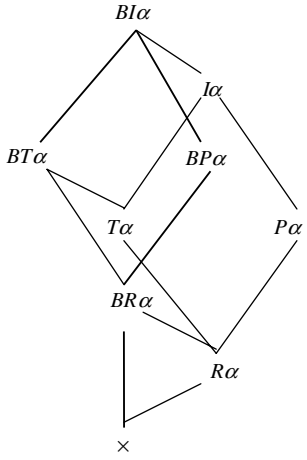

Fig. 2. Order of the semantic link relationship.

3. Rules

3.1. Derivation rules on semantic links

The logical relationships between different semantic links can be described as rules. According to Definition 1, a set of rules can be formed as shown in Table 1 where ',' denotes logical AND. These rules reflect a kind of reuse knowledge between components. Rule1–8 represent the transitive characteristic of the semantic links. Rule9–16 represent the implication relationship between semantic links. The transitivity rules and implication rules enable different semantic links to be connected for derivation. Rule1–16 can be proved according to Definition 1. These are basic rules that can be used to derive more rules. Rule17–20 reflect the logical deduction relationship among semantic links. RA-Rule17–22 can be easily proved according to Definition 1 or the basic rule set. In Rule23–24, $C_1 \text{—}\sim\sim\text{→} C_2$ means that if C_1 is used then C_2 is usually used. $C_1 \text{—}\sim\sim\text{→} (C_2, C_3)$ means that C_1 is used then C_2 andC_3 are usually used together.

As an example, we provide the following proof.

Proof of Rule18. Since $C_1 \text{—}BI\alpha\text{→} C_2 \Rightarrow C_1 \text{—}BT\alpha\text{→} C_2$ (by Rule10), we have the following deduction: $C_1 \text{—}BI\alpha\text{→} C_2$, $C_2 \text{—}BT\alpha\text{→} C_3 \Rightarrow C_1 \text{—}BT\alpha\text{→} C_2$, $C_2 \text{—}BT\alpha\text{→} C_3 \Rightarrow C_1 \text{—}BT\alpha\text{→} C_3$ (by Rule3). □

More rules can be formed based on Definition 1 or can be formed by deducing them from existing rules. These rules can be connected for reasoning purpose. Let $\beta_1\alpha_1$ and $\beta_2\alpha_2$ be two elements of the semantic factor set, and $\beta_1\alpha_1 \cdot \beta_2\alpha_2$ denote the connection of $\beta_1\alpha_1$ and $\beta_2\alpha_2$. The concept of the meaningful connection is defined as follows.

Definition 2. The connection of $\beta_1\alpha_1$ and $\beta_2\alpha_2$ is said to be meaningful if: (1) there exists $\beta_3\alpha_3$ in the factor set and a rule $C_1 \text{—}\beta_1\alpha_1\text{→} C_2$, $C_2 \text{—}\beta_2\alpha_2\text{→} C_3 \Rightarrow C_1 \text{—}\beta_3\alpha_3\text{→} C_3$; and, (2) if $\alpha_1 \leqslant \alpha_2$ then $\alpha_3 = \alpha_1$, if $\alpha_2 \leqslant \alpha_1$ then $\alpha_3 = \alpha_2$, and if $\alpha_1 = \alpha_2$ then $\alpha_3 = \alpha_1 = \alpha_2$.

Table 1
Semantic link rules

No.	Rules	Classification
Rule1	$C_1 \text{—}BI\alpha\text{→}\ C_2,\ C_2 \text{—}BI\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}BI\alpha\text{→}\ C_3$	Transitive
Rule2	$C_1 \text{—}I\alpha\text{→}\ C_2,\ C_2 \text{—}I\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}I\alpha\text{→}\ C_3$	Transitive
Rule3	$C_1 \text{—}BT\alpha\text{→}\ C_2,\ C_2 \text{—}BT\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}BT\alpha\text{→}\ C_3$	Transitive
Rule4	$C_1 \text{—}T\alpha\text{→}\ C_2,\ C_2 \text{—}T\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}T\alpha\text{→}\ C_3$	Transitive
Rule5	$C_1 \text{—}BP\alpha\text{→}\ C_2,\ C_2 \text{—}BP\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}BP\alpha\text{→}\ C_3$	Transitive
Rule6	$C_1 \text{—}P\alpha\text{→}\ C_2,\ C_2 \text{—}P\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}P\alpha\text{→}\ C_3$	Transitive
Rule7	$C_1 \text{—}BR\alpha\text{→}\ C_2,\ C_2 \text{—}BR\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}BR\alpha\text{→}\ C_3$	Transitive
Rule8	$C_1 \text{—}\times\text{→}\ C_2,\ C_2 \text{—}\times\text{→}\ C_3 \Rightarrow C_1 \text{—}\times\text{→}\ C_3$	Transitive
Rule9	$C_1 \text{—}BI\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}I\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}T\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_2$	Implication
Rule10	$C_1 \text{—}BI\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}BT\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}BR\alpha\text{→}\ C_2$	Implication
Rule11	$C_1 \text{—}BI\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}BP\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}BR\alpha\text{→}\ C_2$	Implication
Rule12	$C_1 \text{—}BT\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}T\alpha\text{→}\ C_2$	Implication
Rule13	$C_1 \text{—}BP\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}P\alpha\text{→}\ C_2$	Implication
Rule14	$C_1 \text{—}BR\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_2$	Implication
Rule15	$C_1 \text{—}I\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}P\alpha\text{→}\ C_2 \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_2$	Implication
Rule16	$C_1 \text{—}T\alpha\text{→}\ C_2,\ C_2 \text{—}P\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Implication
Rule17	$C_1 \text{—}R\alpha\text{→}\ C_2,\ C_2 \text{—}\times\text{→}\ C_3 \Rightarrow C_1 \text{—}\times\text{→}\ C_3$	Deduction
Rule18	$C_1 \text{—}BI\alpha\text{→}\ C_2,\ C_2 \text{—}BT\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}BT\alpha\text{→}\ C_3$	Deduction
Rule19	$C_1 \text{—}BI\alpha\text{→}\ C_2,\ C_2 \text{—}BP\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}BP\ \alpha\text{→}\ C_3$	Deduction
Rule20	$C_1 \text{—}I\alpha\text{→}\ C_2,\ C_2 \text{—}R\alpha\text{→}\ C_3 \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Deduction
Rule21	$C_3 \text{—}T\alpha\text{→}\ C_1,\ C_3 \text{—}T\alpha\text{→}\ C_2 \Rightarrow C_3 \text{—}R\alpha\text{→}\ C_1,\ C_3 \text{—}R\alpha\text{→}\ C_2$	Deduction
Rule22	$C_3 \text{—}P\alpha\text{→}\ C_1,\ C_3 \text{—}P\alpha\text{→}\ C_2 \Rightarrow C_3 \text{—}R\alpha\text{→}\ C_1,\ C_3 \text{—}R\alpha\text{→}\ C_2$	Deduction
Rule23	$C_1 \text{—}\sim\sim\text{→}\ C_2,\ C_2 \text{—}\sim\sim\text{→}\ C_3 \Rightarrow C_1 \text{—}\sim\sim\text{→}\ C_3$	Deduction
Rule24	$C_1 \text{—}\sim\sim\text{→}\ C_2,\ C_2 \text{—}\sim\sim\text{→}\ (C_3, C_4) \Rightarrow C_1 \text{—}\sim\sim\text{→}\ (C_3, C_4)$	Deduction

With Definition 2, the transitivity rule: $C_1 \text{—}\beta_1\alpha_1\text{→}\ C_2$, $C_2 \text{—}\beta_2\alpha_2\text{→}\ C_3 \Rightarrow C_1 \text{—}\beta_3\alpha_3\text{→}\ C_3$ can be represented as $\beta_1\alpha_1 \cdot \beta_2\alpha_2 \Rightarrow \beta_3\alpha_3$, i.e., '$\beta_1\alpha_1$ connects with $\beta_2\alpha_2$' implies $\beta_3\alpha_3$. If we represent the implication rule: $C_1 \text{—}\beta_1\alpha_1\text{→}\ C_2 \Rightarrow C_1 \text{—}\beta_2\alpha_2\text{→}\ C_2$ as $\beta_1\alpha_1 \Rightarrow \beta_2\alpha_2$, then we can produce the following lemma by summarizing the rules of Table 1.

Lemma 2. *Let $\beta_1\alpha_1 \cdot \beta_2\alpha_2$ be a meaningful connection of two semantic links. If $\beta_1\alpha_1$ is weaker than $\beta_2\alpha_2$ (i.e., $\beta_1\alpha_1 \leqslant \beta_2\alpha_2$), then $\beta_1\alpha_1 \cdot \beta_2\alpha_2 \Rightarrow \beta_1\alpha_1$. If $\beta_1\alpha_1$is the same as $\beta_2\alpha_2$, then either $\beta_1\alpha_1 \cdot \beta_2\alpha_2 \Rightarrow \beta_1\alpha_1$ or $\beta_1\alpha_1 \cdot \beta_2\alpha_2 \Rightarrow \beta_2\alpha_2$ holds.*

Proof. According to Lemma 1, for any two semantic links $\beta_1\alpha_1$ and $\beta_2\alpha_2$ there exists a semantic factor $\beta_3\alpha_3$, such that $\beta_1\alpha_1 \leqslant \beta_3\alpha_3$, $\beta_2\alpha_2 \leqslant \beta_3\alpha_3$, and $\beta_1\alpha_1 \cdot \beta_2\alpha_2 \Rightarrow \beta_3\alpha_3$, and satisfies: (1) if $\beta_1\alpha_1 \leqslant \beta_2\alpha_2$ then $\beta_3\alpha_3 = \beta_1\alpha_1$ holds; and, (2) if $\beta_1\alpha_1 = \beta_2\alpha_2$ then $\beta_3\alpha_3 = \beta_1\alpha_1 = \beta_2\alpha_2$ holds. □

3.2. Component modification rules

The function of a component can be adapted to help solve similar problems by modifying its edge or node. Such a modification concerns the operations on nodes and edges. We use $C_1 = E(C_2)$ and $C_1 = N(C_2)$ to denote: "C_1 is formed by a modification on the edges of C_2" and "C_1 is formed by a modification on the nodes of C_2" respectively. Any edge modification or node modification may change the behaviour of the component. The edge operations concern: (1) append an edge; (2) delete an edge; and, (3) modify an edge. According to Definition 1, we have the edge modification rules E-Rule1-8 as shown in Table 2. They imply that the edge operations on the ancestor of a semantic link cannot make the semantic link stronger. E-Rule9-11 can be proved by E-Rule1-8. As an example, we prove E-Rule9 as follows.

Table 2
Edge operation rules

No.	Rules	Operation
E-Rule1	$C_1 \text{—}R\alpha\text{→}\ C_2,\ C_2 = E(C_3) \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Edge operation
E-Rule2	$C_1 \text{—}\times\text{→}\ C_2,\ C_2 = E(C_3) \Rightarrow C_1 \text{—}\times\text{→}\ C_3$	Edge operation
E-Rule3	$C_1 \text{—}I\alpha\text{→}\ C_2,\ C_2 = EA(C_3) \Rightarrow C_1 \text{—}T\alpha\text{→}\ C_3$	Add an edge
E-Rule4	$C_1 \text{—}T\alpha\text{→}\ C_2,\ C_2 = EA(C_3) \Rightarrow C_1 \text{—}T\alpha\text{→}\ C_3$	Add an edge
E-Rule5	$C_1 \text{—}P\alpha\text{→}\ C_2,\ C_2 = EA(C_3) \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Add an edge
E-Rule6	$C_1 \text{—}I\alpha\text{→}\ C_2,\ C_2 = ED(C_3) \Rightarrow C_1 \text{—}P\alpha\text{→}\ C_3$	Delete an edge
E-Rule7	$C_1 \text{—}T\alpha\text{→}\ C_2,\ C_2 = ED(C_3) \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Delete an edge
E-Rule8	$C_1 \text{—}P\alpha\text{→}\ C_2,\ C_2 = ED(C_3) \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Delete an edge
E-Rule9	$C_1 \text{—}I\alpha\text{→}\ C_2,\ C_2 = EM(C_3) \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Modify an edge
E-Rule10	$C_1 \text{—}T\alpha\text{→}\ C_2,\ C_2 = EM(C_3) \Rightarrow C_1 \text{—}P\alpha\text{→}\ C_3$	Modify an edge
E-Rule11	$C_1 \text{—}P\alpha\text{→}\ C_2,\ C_2 = EM(C_3) \Rightarrow C_1 \text{—}R\alpha\text{→}\ C_3$	Modify an edge

Table 3
Node operation rules

No.	Rules	Operation
N-Rule1	$C_1 —R\alpha\rightarrow C_2,\ C_2 = N(C_3) \Rightarrow C_1 —R\alpha\rightarrow C_3$	Node operation
N-Rule2	$C_1 —\times\rightarrow C_2,\ C_2 = N(C_3) \Rightarrow C_1 —\times\rightarrow C_3$	Node operation
N-Rule3	$C_1 —I\alpha\rightarrow C_2,\ C_2 = NA(C_3) \Rightarrow C_1 —T\alpha\rightarrow C_3$	Add a node
N-Rule4	$C_1 —T\alpha\rightarrow C_2,\ C_2 = NA(C_3) \Rightarrow C_1 —T\alpha\rightarrow C_3$	Add a node
N-Rule5	$C_1 —P\alpha\rightarrow C_2,\ C_2 = NA(C_3) \Rightarrow C_1 —R\alpha\rightarrow C_3$	Add a node
N-Rule6	$C_1 —I\alpha\rightarrow C_2,\ C_2 = ND(C_3) \Rightarrow C_1 —P\alpha\rightarrow C_3$	Delete a node
N-Rule7	$C_1 —T\alpha\rightarrow C_2,\ C_2 = ND(C_3) \Rightarrow C_1 —P\alpha\rightarrow C_3$	Delete a node
N-Rule8	$C_1 —P\alpha\rightarrow C_2,\ C_2 = ND(C_3) \Rightarrow C_1 —P\alpha\rightarrow C_3$	Delete a node

Proof of E-Rule9. Since $C_2 = EM(C_3)$ can be regarded as the combination of $C_2 = ED(C_3)$ and $C_2 = EA(C_3)$, we have the following deduction: $C_1 —I\alpha\rightarrow C_2$, $C_2 = EM(C_3) \Rightarrow C_1 —I\alpha\rightarrow C_2$, $C_2 = ED(C_3)$, $C_2 = EA(C_3) \Rightarrow C_1 —P\alpha\rightarrow C_2$, $C_2 = EA(C_3) \Rightarrow C_1 —R\alpha\rightarrow C_3$ (by E-Rule 5–6). Thus E-Rule9 holds. □

The node operations include: (1) append a node; (2) delete a node; and, (3) modify a node. Any node operation may cause a modification on the edges related to the node. The node operations relate to the dependence relationships among the nodes of the components to be operated. The node operation rules are shown in Table 3. They imply that any node operation on the ancestor of a semantic link cannot make the semantic link stronger.

4. Inexact semantic link and rule reasoning

4.1. Inexact semantic link

Difference exists between descendants that reuse a common ancestor with the same type of semantic link. To reflect such a difference, we attach a similarity degree ($sd \in [0,1]$) to the semantic link. The larger sd, the more the descendant is similar to the ancestor, i.e., a descendant with a larger sd takes after more behaviours of its ancestor than that with a smaller sd. We extend the semantic link notation $C' —v\rightarrow C$ to $C' —(v,sd)\rightarrow C$, which means that C' takes after C with the semantic factor v and the similarity degree sd. The inexact semantic link is a refinement of the semantic link.

A number of similarity measurement approaches was suggested (Levene and Poulovassilis, 1991; Zadeh, 1971). The similarity degree between two components herein depends on their large-granularity nodes. More common nodes two components share, the larger the similarity degree they have. Let N and N' be the node sets of C and C' respectively, $|N \cap N'|$ be the number of the common node pairs between two components, and $|N \cup N'|$ be the total amount of the nodes included in C and C'. The similarity degree between C and C' can be computed as follows: $sd = \varphi(|N \cap N'|/(|N \cup N'| - |N \cap N'|))$, where $\varphi(x)$ is a function that maps x into $[0,1]$. A simple way to compute sd is: $sd = 2 * |N \cap N'|/|N \cup N'|$. If the nodes of the components are also components, then $|N \cap N'|$ represents the number of matched node pairs, which can be determined by whether the similarity degree between two matched components is bigger than a predefined threshold $\delta \in (0,1)$ or not.

4.2. Inexact rule reasoning

Inexact rule reasoning is a forward linking between rules. Let EOS and NOS be the edge operation set and the node operation set respectively, $C_1 —\gamma_1\rightarrow C_2$ and $C_3 —\gamma_2\rightarrow C_4$ are two semantic links. Rule reasoning is used to match C_2 and C_3, such that $C_1 —\gamma_3\rightarrow C_4$, where γ_3 is determined according to the following three cases:

Case 1: Link rule reasoning. Let $\gamma_i = (v_i, sd_i)$ for $i = 1, 2, 3$.
If $v_1 \cdot v_2 \Rightarrow v_3$, then $sd_3 = \rho(sd_1, sd_2)$, where ρ satisfies: $sd_3 \leqslant sd_1$ and $sd_3 \leqslant sd_2$.

Case 2: Operation rule reasoning.
(1) If $\gamma_1 \in EOS$, $\gamma_2 \in EOS$, then $\gamma_3 \in EOS$ and $\gamma_1 \cdot \gamma_2 \Rightarrow \gamma_3$; and,
(2) If $\gamma_1 \in NOS$, $\gamma_2 \in NOS$, then $\gamma_3 \in NOS$ and $\gamma_1 \cdot \gamma_2 \Rightarrow \gamma_3$.

Case 3: Hybrid reasoning.
If $\gamma_1 = (v_1, sd_1)$, $\gamma_2 \in EOS \cup NOS$, and $\gamma_1 \cdot \gamma_2 \Rightarrow \gamma_3$, then $\gamma_3 = (v_3, sd_3)$, where $v_3 \leqslant v_1$, $sd_3 = k \times sd_1$, and $k \in (0,1)$.

The inexact semantic link provides a natural way to solve the "reuse conflict" issue when one component directly or indirectly takes after another with different semantic factors. For example, we may have the following two kinds of reasoning: (1) $C_0 —T\alpha_1\rightarrow C_1$, $C_1 —T\alpha_2\rightarrow C_3 \Rightarrow C_0 —T\alpha_3\rightarrow C_3$; and, (2) $C_0 —P\alpha_1\rightarrow C_1$, $C_1 —T\alpha_2\rightarrow C_3 \Rightarrow C_0 —R\alpha_3\rightarrow C_3$. The two reasoning results cause two kinds of semantic links to occur between the same pair of components: C_0 and C_3. With the inexact semantic link, two different kinds of semantic links reflect the reuse relationship between two components from different views and with different similarity degrees. A weaker semantic link reflects a smaller similarity degree between two components, while a stronger semantic link reflects a larger similarity degree between two components

(according to Definition 1). Generally we have the following proposition:

Proposition 1. *Let* $C_0 —(v_1, sd_1)→ C_1$ *and* $C_0 —(v'_1, sd'_1) → C_1$ *be two semantic links between* C_0 *and* C_1. *If* $v_1 \leqslant v'_1$, *then* $sd_1 \leqslant sd'_1$ *holds*; *or*, *if* $v'_1 \leqslant v_1$, *then* $sd'_1 \leqslant sd_1$ *holds.*

5. Component overlay

Based on the inexact semantic link, we can construct a multi-level component overlay to support flexible component reuse.

5.1. Basic structure model

The basic structure model of the component overlay consists of three levels top-down: a component level, a model level, and a function level as shown in Fig. 3. The component level consists of a group of component networks, each of which is constituted by a set of components together with the inexact semantic link. The model level consists of a group of model networks, each of which is constituted by a set of models together with the inexact semantic link. The inexact semantic link between models can be similarly defined as the semantic link between components by mapping the model implementation structure into the component implementation structure. Similar to the notation of the inexact component semantic link, "$M_i —(v, sd)→ M_j$" is used to denote that M_i takes after M_j with the semantic factor v and with a similarity degree sd. The model is implemented by the composition of functions. The function level consists of function specialization graphs (*FSG*) within which nodes are functions, and edges are function specialization relationships as defined in (Zhuge, 1998). The function specialization relationship does not exist among the nodes belonging to different *FSG*s.

The high level of the component overlay has a small number of large-granularity nodes. The low-level has a large number of small-granularity nodes. The high level node is implemented by a network constructed out of the low level nodes and the edges among them. Take Fig. 3 for example, C_4 at the component level is implemented by a network that consists of the model set $\{M_1, M_2, M_5\}$ of the model level and the edge set $\{\mathrm{DD}, \mathrm{FD}\}$. The component overlay structure presents the general information at the high level and the detail information at the low level, so the structure supports the macro top-down refinement retrieval strategy.

Let *CS* be the set of components at the component level, *MS* be the set of models at the model level, *FS* be the set of functions at the function level, *CRA and MRA* be the inexact component semantic link and the model semantic links, *SpecR* be the specialization relationship on *FS*, and '$\leqq$' be the order among different levels. The component overlay *CO* can be described as follows:

$$CO = \langle \{\langle CS, CRA\rangle, \langle MS, MRA\rangle, \langle FS, SpecR\rangle\}, \leqq\rangle,$$

which satisfies:

(1) $\langle FS, SpecR\rangle \leqq \langle MS, MRA\rangle \leqq \langle CS, CRA\rangle$;

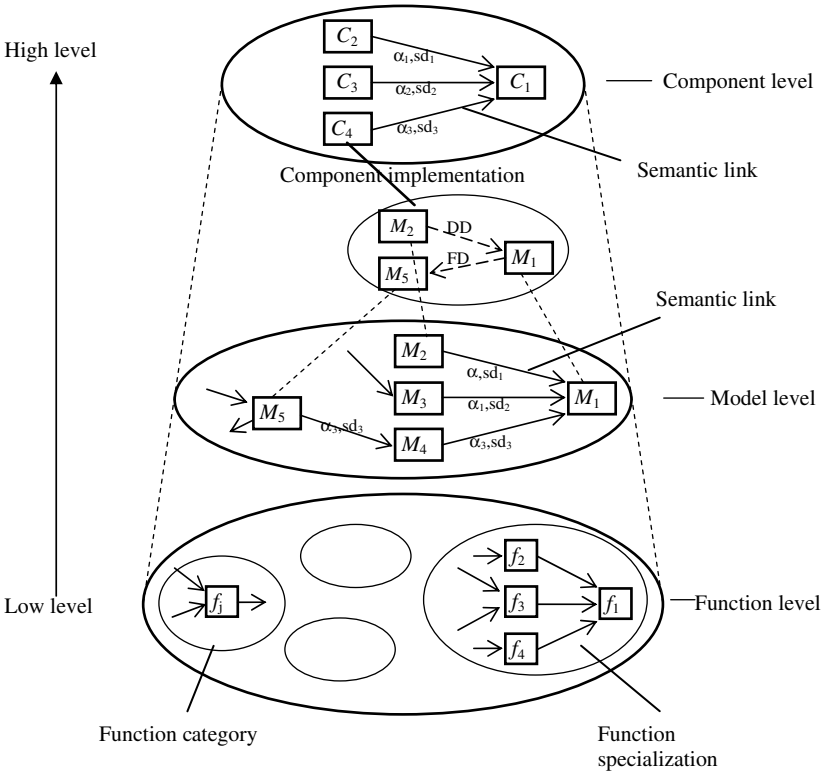

Fig. 3. Basic structure of the component overlay.

(2) For any $C \in CS$, C is implemented by a subset of MS, and for any $M \in MS$, M is implemented by a subset of FS; and,
(3) Any nodes of the CO are uniformly specified by the following frame structure:
NodeID: [Name: CodeString(level);
Signature: $InputType \rightarrow OutputType$;
Type: $\langle N, E, R \rangle | \ SimpleType$;
Behaviour: $TextDescription$;
Environment:
$EnvironmentalVariableDescription$].

5.2. Basic operations

The basic operations of the component overlay include "retrieve", "create", "append", and "delete". We herein mainly introduce the component retrieval operation and creation operation. Component retrieval is to search for the required component in terms of users' retrieval requirements, which will be refined during searching. The searching mechanism needs to check the similarity degree and the semantic link between the requirement (C_Q) and the current node (C_c), and then to select the next visiting node (C_n) according to the heuristic rules in Table 4. If there exists more than one candidate node that satisfy the heuristic rules, then the retrieval mechanism will select the one that takes after the required component with a bigger similarity degree. The user interface of the retrieval operation is an SQL-like command like the model retrieval interface as defined in (Zhuge, 1998). The similarity degree can be included into the condition of these commands. For example, "SELECT ALL FROM CO.CLevel WHERE * -(v_i, sd_i)-> C" is used to select all the components that take after C with a semantic link v_i and with a similarity degree sd_i from the component overlay CO at the component level (CLevel).

The creation of a new component is to define its implementation and establish the semantic link between the new component and existing components. The function $Create(n, \text{Location})$ is responsible for creating a node, n, at the given location (e.g., level). For example, $Create(C, \text{CO.CLevel.Cat1})$ is used to create a component C in the category "Cat1" at the component level. The "create" function is implemented through reuse at three levels: (1) if there exists a reusable component at the component level, then the new component can be created by reusing the existing component through the inexact semantic link; (2) if there does not exist a reusable component, but there exist some models at the model level that can constitute the new component, then the new component can be created by composing the existing models; and, (3) if there does not exist a reusable component at the component level nor the required models at the model level but there exists some functions at the function level that can be used to constitute the required models, then use these functions to compose the required new models, which can be further used to compose new components.

Table 4
Heuristic rules for component retrieval

RuleNo.	Heuristic rules
H-Rule1	If C_c —$T\alpha$→ C_Q then select C_n that satisfies C_n —$P\alpha$→ C_c
H-Rule2	If C_c —$P\alpha$→ C_Q then select C_n that satisfies C_n —$T\alpha$→ C_c
H-Rule3	If C_c —$R\alpha$→ C_Q then select C_n according to the following priority order: $\langle C_n$ —$T\alpha$→ C_c, C_n —$R\alpha$→ C_c, C_n —$P\alpha$→ $C_c\rangle$
H-Rule4	If C_c —$I\alpha$→ C_Q then select C_n that satisfies C_n —$I\alpha$→ C_c

Other operations include: (1) $DelN(n, \text{Location})$, the function for deleting a node n at the given location; (2) $DelE(n, n', t, \text{Location})$, the function for deleting an edge with type t between node n and n' at the given location; (3) $AppN(n, \text{Location})$, the function for appending a node n at the given location; (4) $AppE(n, n', t, \text{Location})$, the function for appending an edge with type t between node n and n' at the given location; (5) $ModN(n, \text{Location})$, the function for modifying a node n at the given location. The edge modification operation can be realized by merging the edge operation $DelE$ with the node operation $AppN$.

5.3. Assessment

We now examine the effectiveness and the efficiency of the component overlay CO with the following three assumptions: (1) MB is a model base containing the same set of models as the model level of the CO; (2) every component of the CO is constructed by its model-level models; and, (3) MSR denotes the model set required by a problem-solving process. According to the assumptions and the structure of the component overlay CO, we have: if MSR is a subset of the MB then MSR is also a sub-set of the model level of CO, vice versa. So the model level of the CO is as effective as the MB for supporting the same problem-solving process. Hence, we have the following lemma.

Lemma 3. *The component overlay is as effective as the model base for supporting the same problem-solving process.*

The component overlay provides the structural information and the semantic link for the search mechanism to enhance its efficiency. We assumedly make an index on the model base MB according to the behaviour categories of the models. The retrieval efficiency of the MB with the index is higher than that without the index. Now, we analogy the CO to the MB with the index by mapping the component level of the CO into the index of the MB, and by mapping the model level of the CO into the MB. Intuitively, the retrieval efficiency of the CO is not lower than that of MB with the index, so we have the following lemma.

Lemma 4. *The retrieval efficiency of the component base CO is higher than that of the model base MB for supporting the same problem-solving process.*

Proof.

Case 1: All the models in the *MSR* are available in the *MB*. In this case, all the models in the *MSR* are also available at the model level of the *CO* according to the assumption. So the retrieval efficiency of the *CO* is the same as that of the *MB* with index.

Case 2: A model M_k in *MSR* is not available in the *MB* but can be composed by several existing models. In this case, M_k should be available at the component level of the *CO* according to the assumption. The retrieval mechanism of the *CO* does not need to search for the models one by one at the model level of the *CO* for composing M_k, but the retrieval mechanism of the *MB* needs to search for the required models one by one for composing M_k. So the retrieval efficiency of the *CO* is higher than that of *MB* with the index in this case.

Case 3: A model M_k in *MSR* does not exist in the *MB* and cannot be composed by existing models. In this case, the retrieval mechanism of the *MB* still needs to check all the existing models. The implementation relationship of a component enables the retrieval mechanism of the *CO* to search for only the component level, which contains fewer nodes than that of the model level. So the retrieval efficiency of the *CO* is higher than that of *MB* with the index in this case.

Summarizing above three conclusions, we can reach that the retrieval efficiency of the *CO* is higher than or equal to that of the *MB* with the index. Considering the fact that the retrieval efficiency of the *MB* with the index is higher than that without the index, so Lemma 4 holds. □

The semantic link can further enhance the search efficiency by increasing the width of the component overlay hierarchy (i.e., the average number of the children of every component node).

More component overlays can be constructed, but, too many levels will be a burden for the maintenance mechanism. The proper number of the component levels depends on the total number of the repository components and the proper number of the components arranged at the basic component level. A practicable way to determine the number of the component levels is to use the experience approach for arranging the multi-level module structure of a system in the structured analysis and design method.

The maintenance cost of the component overlay is higher than that of the conventional model base because the maintenance mechanism of the component overlay

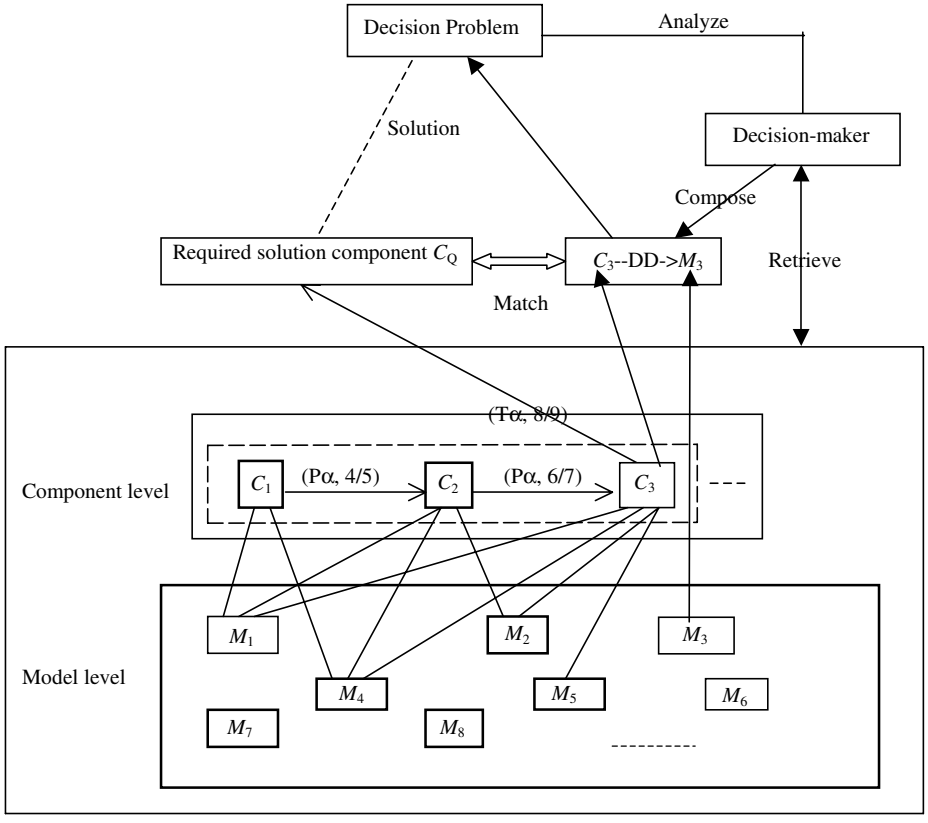

Fig. 4. Decision-making supported by the component base and model base. *Legend*: M_1: investment-profit analysis model, M_2: long-term production-planning model, M_3: sales-planning model, M_4: sales-forecasting model, M_5: production-scheduling model, M_6: storage management model, M_7: cost-effective model, M_8: assessment model, the implementation of C_1: M_4 —DD→ M_1, the implementation of C_2: M_4 —DD→ M_1 —DD→ M_2, the implementation of C_3: M_4 —DD→ M_1 —DD→ M_2 —DD→ M_5, the required component C_Q: M_4 —DD→ M_1 —DD→ M_2—DD→ M_5 —DD→ M_3.

needs to maintain the relationships of multiple levels, while the traditional model base just needs to maintain the model set by conventional file management system. In practice, the maintenance can be done off line or by other processes or services, so the efficiency of the problem-solving platform mainly depends on the search efficiency.

6. Application in building problem-oriented component base

6.1. Overview

The proposed approach was applied to the establishment of the problem-oriented component base system PROMBS. At the first stage of the system development, a model base system RMMBS (Zhuge, 1998) was developed to improve the traditional FORTRAN-based mathematical software library (IMSL Math/Library) in two aspects: (1) provide an inexact query mechanism for the users (especially the beginners) in the scientific computing field to flexibly retrieve mathematical models (which are coded in different languages); and, (2) increase the reusability of the models by establishing multiple 'inheritance' among the models at the signature level. At the second stage, the system RMMBS was upgraded to the system PROMBS (Zhuge, 2000), which can provide a problem-oriented high-level description language POL for users to describe the solutions to problems with the support of the repository components. A translator mechanism is responsible for translating the solution description into the model-based programs, where the specification-level models are mapped into the implementation-level models (coded in 3GL) during the translation process. At the third stage, the system PROMBS was evolved into the component base system that supports flexible component reuse in solving decision problems (see Fig. 4). A large-granularity component level was established above the model base that was developed at the first stage. We upgraded the multiple 'inheritance' relationship to the inexact semantic link, and incorporated the related rules and rule reasoning mechanism into the retrieval mechanism.

6.2. Component reuse example

6.2.1. Background description

Managers need to make production and operation management decisions according to the market demands and also need to adjust these decisions according to the change of market. Before making decision to invest a new project, managers need to make market forecast and profit forecast so as to choose the best project. The market forecast model and the profit forecast model should be composed as a component for later use. Having made their investment decisions they need to make long-term production plans, storage management plans, and sales plans. While they are making production plans, market change may occur, so they need to keep updating the market analysis and the profit analysis so as to adjust their investment decisions. At this time they can reuse the existing component rather than composing the solution from scratch.

6.2.2. Component overlay description

The model level includes the following models or model categories.

(1) Analysis model category, including the investment profit analysis model and the risk analysis model.
(2) Production planning model category, including the short-term production planning model, the long-term production planning model, and the production scheduling model.
(3) Sales model category, including the sales planning model and the sales forecasting model.
(4) Storage management model.
(5) Cost-effect model.
(6) Assessment model.
(7) Capacity evaluation model.

The component level consists of three components: C_1, C_2 and C_3. Their implementations are described as the legend in Fig. 5. The component overlay is graphically described in Fig. 5. We herein assume that the required component (C_Q) is represented as: C_Q: M_4 —DD→ M_1 —DD→ M_2 —DD→ M_5 —DD→ M_3. Since there does not exist a component in the component overlay that exactly matches the requirement, users (decision-makers) can use the inexact retrieval mechanism to find a reusable component.

6.2.3. Inexact reuse example

The users can use 'SELECT ALL FROM CO.CLevel.Cat1 WHERE $*$ -Pα-> CQ AND $\mathrm{sd}(\mathrm{CQ}, *) \geqslant 0.85$' to express that the candidate components (denoted as '*') partially take after the required component and that the similarity degrees should be bigger than or equal to 0.85. The retrieval mechanism just needs to check one of the three components, and then the retrieval mechanism can find the proper component by rule reasoning.

Now we examine the following three cases that the retrieval mechanism may encounter.

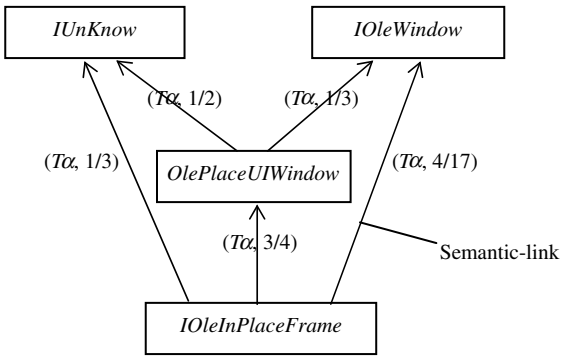

Fig. 5. Semantic link hierarchy on COM components.

(1) If component C_3 is the first one to be checked, then the target C_3 has been found because the inexact semantic link exists: C_3 —$(P\alpha, 8/9)$→ C_Q, and the similarity degree 8/9 is bigger than the required 0.85.
(2) If component C_2 is the first one to be checked, then the retrieval mechanism gets the semantic link: C_2 —$(P\alpha, 6/8)$→ C_Q. So the rule reasoning mechanism needs to find a component C that satisfies: C_2 —$(P\alpha, sd)$→ C, where sd is the closest to 6/8. Since C_2 —$(P\alpha, 6/7)$→ C_3 and C_3 is the last node, C_3 satisfies the requirement.
(3) If component C_1 is the first one to be checked, then the retrieval mechanism gets the semantic link: C_1 —$(P\alpha, 4/7)$→ C_Q. The rule reasoning mechanism needs to find a component C that satisfies: C_1 —$(P\alpha, sd)$→ C, where sd is the closest to 4/7. By rule reasoning: C_1 —$(P\alpha, 4/5)$→ C_2, C_2 —$(P\alpha, 6/7)$→ $C_3 \Rightarrow C_1$ —$(P\alpha, 4/7)$→ C_3, the retrieval mechanism can find the suitable component C_3. Hence the inexact semantic link reasoning can support the flexible component reuse.

The inexact rule reasoning mechanism can also explain the retrieval result based on the proposed rules. For example, it can list all the candidate components with the similarity degrees and the semantic link between them.

7. Experiment for building common software component overlay

The proposed approach was used to build a COM interface component overlay, which consists of a software component level and a method level (in COM context). A software component herein is implemented by a set of methods at the method level. The following are four components:

C_1: [Name: *IUnKnow*;
Implementation: ⟨{QueryInterface, AddRef, Release}, {}, {}⟩];
C_2: [Name: *IOleWindow*;
Implementation: ⟨{GetWindow, ContextSensitiveHelp}, {}, {}⟩];
C_3: [Name: *IOlePlaceUIWindow*;
Implementation: ⟨{QueryInterface, AddRef, Release, GetWindow, ContextSensitiveHelp, GetBorder, RequestBorderSpace, SetBorderSpace, SetActiveObject}, {}, {}⟩];
C_4: [Name: *IOleInPlaceFrame*;
Implementation: ⟨{QueryInterface, AddRef, Release, GetWindow, ContextSensitiveHelp, GetBorder, RequestBorderSpace, SetBorderSpace, SetActiveObject, InsertMenus, SetMenu, RemoveMenus, SetStatusText, EnableModeless, TranslateAccelerator}, {}, {}⟩].

The inexact semantic link among the four components is shown in Fig. 5. Users can use the query 'SELECT ALL WHERE $\mathrm{sd}(C_Q = \{IOlePlaceUIWindow\}, *) \geqslant 0.75$' to retrieve all the components that match C_Q with the similarity degree bigger than or equal to 0.75. The retrieval mechanism carries out the searching and then returns the query result: $\{IOleInPlaceFrame\}$. If the query condition is relaxed to 'SELECT ALL WHERE $\mathrm{sd}(C_Q = \{IOlePlaceUIWindow\}, *) \geqslant 0.5$', then we will get the query result: $\{IOleInPlaceFrame, IUnKnow\}$. If the query condition is relaxed to 'SELECT ALL WHERE $\mathrm{sd}(C_Q = \{IOlePlaceUIWindow\}, *) \geqslant 0.3$', then we will get the query result: $\{IOleInPlaceFrame, IUnKnow, IOleWindow\}$.

The repository architecture of previous COM interface is classification based and the retrieval mechanism is keyword based. The proposed approach provides a way to improve previous mechanism by establishing inexact semantic links among components and by providing the inexact retrieval mechanism. Besides, a large-granularity component level can be formed based on the existing component level and method level. A flexible reuse at the large-granularity level can be similarly realized. This example implicates that the approach is also suitable for realizing inexact reuse of well-defined software components. For example, we can realize the flexible reuse of software patterns by establishing inexact semantic links among patterns (or the large-granularity pattern components) in a software pattern base. COM and DCOM do not have this technology, but it is useful in Web services.

8. Application in realizing flexible Web service retrieval

With the rapid development of Internet applications and service-oriented business model, component technologies are developing towards distributed Web service paradigm, which aims at providing an open platform for the development, deployment, interaction, and management of globally distributed e-services. The Web standards like SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Definition, Discovery and Integration) are the implementation basis of Web Services.

The key to implement Web services is to discover the appropriate service that matches a demand from large candidate services. So the semantic description of services and matching between services become the key issue. Direct description of the functions of services is one way (Zhuge, 2004), but sometimes the function of a service can be determined by the functions of semantically related services. The proposed approach can be used for describing the semantics between services that can help determine appropriate services and raise the efficiency of service retrieval.

To well-organize services is another aspect that is important to raise the efficiency of service retrieval. The Knowledge Grid is an ideal platform that can publish, share and manage resources including information, knowledge and services across the Internet (Zhuge, 2004). It includes two major components: (1) a Resource Space Model RSM that uniformly specifies and organizes resources in normal

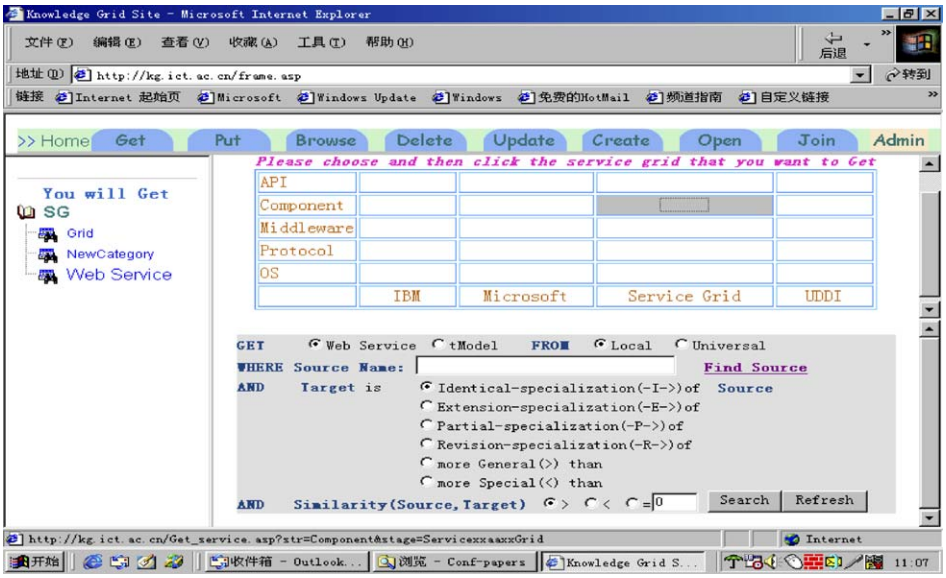

Fig. 6. Interface for flexible retrieval in a service overlay grid.

forms; and, (2) a resource operation mechanism that enables users to conveniently use resources in the resource space. The semantic link and the RSM reflect the classification semantics and the link semantics, which can be used for organizing services in an efficient way.

With reference to the proposed approach, we established the inexact relationships between Web services and between Models, and realized flexible Web service retrieval based on the inexact relationships (Zhuge and Liu, 2004). Fig. 6 shows the operation interface for realizing flexible Web Service retrieval.

9. Synergy normal organization and self-organization

Just as the normalized database guarantees the efficiency and effectiveness of data management, a well-organized component overlay ensures the efficiency of component reuse. The normalization theory of the Knowledge Grid can be used to normalize the local component repository (Zhuge, 2004). The expansion of various components distributed over the Internet challenges the normalization of a global component repository. The flat, equal and autonomous organization paradigms become increasingly important. To synergy the normal organization and self-organization can raise the efficiency of component management in a dynamic distributed environment. One way to realize such a synergy is to use the P2P network at the bottom for accurately locating components in a large dynamic network, and to use the normal organization in the upper overlay to realize normal organization of resources (Zhuge, 2004).

The current Web is an Internet information overlay that self-organizes Web pages for human browsing by hyperlinks. Intelligent applications also need knowledge and services. The semantic links such as the "cause-effect" and "implication" relationships connect distributed knowledge to form a knowledge overlay. The semantic links such as "sequential" and "similar-to" connect services to form a service overlay. The composition of the underlying overlays forms different types of components. The semantic links such as "is-part-of" and "similar-to" connect components to form the component overlay. Fig. 7 shows a solution to synergy these overlays. The information overlay and the knowledge overlay support the service overlay, which enriches the knowledge overlay and the information overlay during use.

In the future Knowledge Grid environment, information, knowledge and service will take the same form that can dynamically organize themselves according to requirement and evaluation of mutual-benefit (Zhuge, 2004, 2005). Some organizations can be generalized as component for reuse in new cases. The function of a component can be adapted by changing its internal links.

10. Networking autonomous spaces in e-Science Knowledge Grid environment

The notion of components can be promoted from passive reuse (components are selected by an application program or user) to active clustering (components can autonomously cluster to solve a problem). Passive reuse and active clustering effect differently: passive reuse reduces problem-solving cost, while active clustering gains problem-solving ability.

The future interconnection environment has five basic parameters: space, time, structure, relation, and worth (Zhuge, 2005). The environment needs the ability of active clustering—clustering autonomous virtual spaces that can integrate services to provide personalized services. The high-level architecture of the China e-Science Knowledge Grid environment IMAGINE-I has: *research roles*, research *space* and research *resources* (resources are defined by structure). People and agents can play roles of the leader,

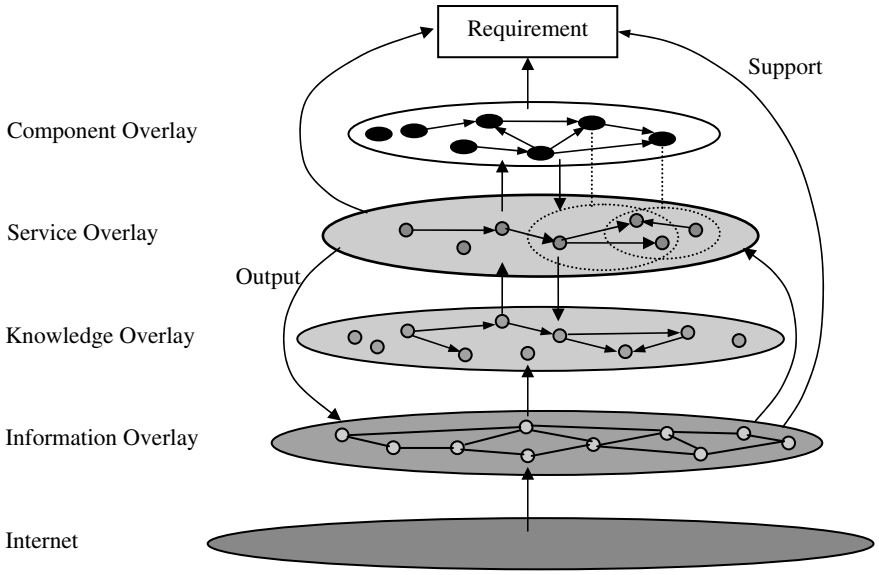

Fig. 7. The synergy of the component overlay, service overlay, knowledge overlay and information overlay.

member, collaborator, supervisor, student, etc. Resources include data, papers, books, tools, instruments, etc. Research spaces are virtually personalized research spaces where researchers can use appropriate on-demand services. Services are dynamically clustered from candidates distributed world-widely.

The architecture of a research space takes after the soft-device (Zhuge, 2002). A new research space can be generated by inheriting from a root space—a software framework, and then personalizing the following items:

(1) User profile, including user's login information and research interests.
(2) Services, researchers' needs in research.
(3) Activities, the user research activities.
(4) Workflows, research processes that order activities.

A resource space together with its users constitutes a *semantically interoperation intelligent unit* that can work and cooperate with others autonomously. Research spaces can dynamically get together to form research communities. A research space can join and leave a community freely. The proposed semantic relationships between components exist between research spaces.

In addition, the following semantic relationships also exist between research spaces:

(1) A —peer→ B, A and B belong to the same community;
(2) A —cooperate→ B, A cooperates with B;
(3) A —supervise→ B, A supervises B in research and development;
(4) A —serve→ B, A serves B with a logistic workflow;
(5) A —share→ B, A shares with B on research resources;
(6) A —help→ B, A helps B in answering questions;
(7) A ∼∼help→ C, A indirectly helps C when A —help→ B and B —help→ C.

The following implication relationship exists between these semantic links:

(1) A —peer→ B ⇒ B —peer→ A;
(2) A —cooperate→ B ⇒ B —cooperate→ A;
(3) A —share→ B ⇒ B —share→ A; and,
(4) A —cooperate→ B ⇒ A —share→ B.

These semantic links connect personalized research spaces to form an autonomous Semantic Grid. Help is usually invoked by queries, and queries can be forwarded from one peer to another to find an appropriate answer. Semantic links can enhance the efficiency of routing among these spaces (Zhuge et al., 2005). Research spaces can play even more active role in research.

Working with such research spaces, researchers discuss, experiment, and document with easily available research resources to share a knowledge flow network. The knowledge flow network together with the underlying Semantic Grid form a live Knowledge Grid.

11. Related work and discussion

Semantic links have been enriched by more semantic factors such as implication, cause-effect and similar-to for wider applications. More semantic factors can be defined according to specific application domain. An algebra model for computing semantic links has been developed (Zhuge, 2004).

Inheritance is a way to reuse software. Studies on inheritance concern the fields of OO methodology (Booch, 1993;

Sen, 1997), OOP (Canning et al., 1989; Wegner, 1990), functional programming, type theory and formal semantics (Bradley and Clemence, 1988; Clark, 1995; Cook and Palsberg, 1989). They focus on either the implementation aspects or the behaviour aspects. From the viewpoint of OOP, inheritance is a reuse or an incremental definition mechanism. Four kinds of compatibility of inheritance (i.e., behaviour compatibility, signature compatibility, name compatibility, and cancellation) were suggested in (Wegner, 1990). The 'white box' inheritance and the 'black box' inheritance were discussed in (Edwards, 1977).

Compared with the traditional inheritance concept, the presented semantic link has two main advantages. First, the semantic link relationship has richer semantics than traditional inheritance. Inheritance can be regarded as a kind of semantic link. Second, the definition of the semantic link is language-irrelevant, while the inheritance of the OOP only supports the code-level reuse. The code-level reuse depends on implementation language. For example, a component coded in C++ cannot inherit from a component coded in another language like Java.

The problem of software component reuse has been widely studied (Batory and Malley, 1992; Felice, 1993; Muhanna, 1993; Rajlich and Silva, 1996; Thompson, 1991; Yau and Tsai, 1987; Zaremski and Wing, 1995). Solutions can be classified into three categories: (1) the formal semantic methods like the algebraic, the operational and the denotation semantic methods; (2) the informal methods like information retrieval based on the storage and retrieval context of components, the topological methods, and the knowledge-based methods; and, (3) the combination of the formal and the informal.

The semantic overlay indirectly specifies semantics. It is between formal and informal, and should belong to a 'grey box' approach. On one hand, it is language-irrelevant, so the approach is a black box reuse (below the function level is black). On the other hand, a component definition explodes its implementation structure. So it has the advantages of both 'black box' reuse and 'white box' reuse, and it can overcome the disadvantages of the 'white box' reuse.

Compared with the hyper-graph data model (Levene and Poulovassilis, 1991), the presented component overlay model has two characteristics: (1) components are organized in semantic links, while the hyper-graph data model is based on the value-dependence relationship; and, (2) the component base retrieval is a heuristic searching based on semantic link and rule reasoning, while the retrieval mechanism in the hyper-graph data model is based on the value-based function dependence relationship.

Compared with the fuzzy relationship (Zadeh, 1971), the semantic link relationship is objectively defined according to the similarity degree among the well-defined components, and can be derived according to relevant links. This objectiveness is required by the fact that a user should be clear about whether a component could completely solve a problem (or a sub-problem) when composing the solution. While, the initial fuzzy factor of the fuzzy relationship has to be subjectively given. Besides, the semantic links can serve to refine the relationships among components.

There are three differences between the semantic links and the association rules in the data-mining field (Han and Fu, 1999). First, the former reflects richer semantics among components, while the later is a kind of 'cause-effect' rule among data elements. Second, the discussed object of the former is complex object but the latter is data. The former focuses on solving complex or large-scale problems, while the latter is suitable for supporting small and low-level decisions. Third, the former is a multi-value relationship while the latter is a fuzzy relationship.

To form a more powerful component retrieval approach, the presented approach can be integrated with other related approaches such as: (1) the white-box reuse, e.g., the mechanism of modifying a component to suit the new requirement (Batory et al., 2000); (2) the automatic component generation mechanism (Mili et al., 1997); and, (3) the GUI of the retrieval mechanism, e.g., the mobile GUI that enables computer-illiterate people to use the retrieval mechanism, as well as the active browsing interface that could raise retrieval efficiency (Drummond et al., 2000).

A component can also be a process that can be passively reused, for example, the component-based workflow development (Zhuge, 2003). The component-based workflow development has potential in promoting the efficiency of workflow system development and in increasing the comprehensibility and correctness of the workflow design. A component can also be an active process. An active process component can actively find relevant processes and integrate with each other to form more complex process according to requirements.

12. Conclusion

Intelligent applications in the future interconnection environment need the cooperation of distributed resources and the support from high to low semantic levels. This paper proposes an approach to support intelligent applications by establishing semantic component networking overlays. Flexible component reuse is realized by semantic component retrieval and rule reasoning on semantic link network. The semantic component overlay is effective and efficient. Compared with previous software reuse approaches, the proposed approach is flexible, high-level, experience-based, and language-irrelevant. The proposed approach has potential in realizing flexible reuse of various components such as software components, workflow components, text components, Web services, XML components, semantic link network components, and knowledge components. The synergy of the static component reuse and dynamic component clustering enables the proposed approach to support intelligent applications in the Knowledge Grid environment.

References

Batory, D., Malley, S.O'., 1992. The Design and Implementation of Hierarchical Software Systems with Reusable Components. ACM Trans. Software Eng. Methodol. 1 (4), 355–398.

Batory, D., Chen, G., Robertson, E., Wang, T., 2000. Design wizards and visual programming environments for GenVoca generators. IEEE Trans. Software Eng. 26 (5), 441–452.

Blanning, R.W., 1993. Model management systems: an overview. Decision Support Syst. 9 (1), 9–18.

Boisvert, R.F., Howe, S.E., Kahaner, D.K., 1985. GAMS: a framework for the management of scientific software. ACM Trans. Math. Software 11 (4), 313–355.

Booch, G., 1993. Object-oriented Analysis and Design with Applications. Benjamin/Cummings, Redwood City, CA.

Bradley, G., Clemence, R., 1988. A type calculus for executable modelling languages. IMA J. Math. Manage. 3 (1), 277–291.

Canning, P.S. et al., 1989. Interfaces for strongly-typed object-oriented programming. In: Proceedings of OOPSLA'89, October, New Orleans, pp. 457–467.

Clark, R.G., 1995. Type safety and behavioral inheritance. Inform. Software Technol. 37 (10), 539–545.

Cook, W., Palsberg, J., 1989. A denotational semantics of inheritance. In: Proceedings of OOPSLA'89, October, New Orleans, pp. 433–443.

Dolk, D.R., Konsynski, B.R., 1984. Knowledge representation for model management systems. IEEE Trans. Software Eng. 10 (6), 619–628.

Drummond, C.G., Ionescu, D., Holte, R.C., 2000. A learning agent that assists the browsing of software libraries. IEEE Trans. Software Eng. 26 (12), 1179–1196.

Edwards, S.H., 1977. Representation inheritance: a safe form of 'White Box' code inheritance. IEEE Trans. Software Eng. 23 (2), 83–92.

Felice, P.D., 1993. Reusability of mathematical software: a contribution. IEEE Trans. Software Eng. 19 (8), 835–843.

Han, J., Fu, Y., 1999. Mining multiple-level association rules in large databases. IEEE Trans. Knowledge Data Eng. 11 (5), 798–805.

Hong, S.N., Mannino, M.V., Greenberg, B., 1993. Measurement theoretic representation of large, diverse model bases—the Unified Modelling Language L_U. Decision Support Syst. 10 (3), 319–340.

Lenard, M.L., 1993. An object-oriented approach to model management. Decision Support Syst. 9 (1), 67–73.

Levene, M., Poulovassilis, A., 1991. An object-oriented data model formalised through hypergraphs. Data Knowledge Eng. 6, 205–224.

Mili, R., Mili, A., Mittermeir, R.T., 1997. Storing and retrieving software components: a refinement based system. IEEE Trans. Software Eng. 23 (7), 445–460.

Muhanna, W.A., 1993. An object-oriented framework for model management and DSS development. Decision Support Syst. 9 (2), 217–229.

Novak Jr., G.S., 1997. Software reuse by specialisation of generic procedures through views. IEEE Trans. Software Eng. 23 (7), 401–417.

Rajlich, V., Silva, J.H., 1996. Evolution and reuse of orthogonal architecture. IEEE Trans. Software Eng. 22 (2), 153–157.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., 1991. Object-Oriented Modelling and Design. Prentice-Hall, Englewood Cliffs, NJ.

Sen, A., 1997. The role of opportunism in the software design reuse process. IEEE Trans. Software Eng. 23 (7), 418–436.

Thompson, S., 1991. Type Theory and Functional Programming. Addison-Wesley, Reading, MA.

Wegner, P., 1990. Concepts and paradigms of object-oriented programming. OOPS Messenger 1 (1), 7–87.

Yau, S.S., Tsai, J.J., 1987. Knowledge representation of software components interconnection information for large-scale software modifications. IEEE Trans. Software Eng. 13 (3), 355–361.

Zadeh, L.A., 1971. Similarity relations and fuzzy ordering. Inform. Sci. 3 (3), 177–200.

Zaremski, A.M., Wing, J.M., 1995. Signature matching: a tool for using software libraries. ACM Trans. Software Eng. Methodol. 4 (2), 146–170.

Zhuge, H., 1998. Inheritance Rules for Flexible Model Retrieval. Decision Support Syst. 22 (4), 379–390.

Zhuge, H., 2000. A problem-oriented and rule-based component repository. J. Syst. Software 50, 201–208.

Zhuge, H., 2002. Clustering soft-devices in semantic grid. IEEE Comput. Sci. Eng. 4 (6), 60–62.

Zhuge, H., 2003. Component-based workflow systems development. Decision Support Syst. 35 (4), 517–536.

Zhuge, H., 2004. The Knowledge Grid. World Scientific Publishing Co., Singapore.

Zhuge, H., 2005. The future interconnection environment. IEEE Computer 38 (4), 27–33.

Zhuge, H., Liu, J., 2004. Flexible retrieval of Web services. J. Syst. Software 70 (1–2), 107–116.

Zhuge, H. et al., 2005. Query routing in a Peer-to-Peer semantic link network. Comput. Intelligence 21 (2), 197–216.

Hai Zhuge is the professor and director of the Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences. He leads the China Knowledge Grid Center (http://www.knowledgegrid.net), which owns three branches: China Knowledge Grid Research Group (http://kg.ict.ac.cn), Dunhuang Knowledge Grid Lab, and Hunan Knowledge Grid Lab. The centre employs over 30 young researchers. He is the chief scientist of the China Semantic Grid project—a five-year national basic research program project. He was the keynote speaker at the 2nd International Conference on Grid and Cooperative Computing (GCC03), the 5th and 6th International Conference on Web-Age Information Management (WAIM04 and WAIM05), the 1st International Workshop on Massively Multi-Agent Systems (MMAS04), the 1st International Workshop on Autonomous Intelligent Systems—Agents and Data Mining (AIS-ADM05), the 8th International Conference on Electronic Commerce (ICEC2006), and 1st Asian Semantic Web Conference (ASWC2006). He was the chair of the 2nd International Workshop on Knowledge Grid and Grid Intelligence (KGGI04), the program chair of the 4th International Conference on Grid and Cooperative Computing (GCC05), and the chair of the 1st International Conference on Semantics, Knowledge and Grid (SKG2005) and the program chair of SKG2006. He has organized several journal special issues on Semantic and Knowledge Grid. He is serving as the Area Editor of the *Journal of Systems and Software* and the Associate Editor of *Future Generation Computer Systems*. His current research interests are the models, theories, methods and applications of the future interconnection environment. His monograph *The Knowledge Grid* is the first monograph in this area. He has over sixty papers appeared mainly in leading international journals such as *Communications of the ACM*; *IEEE Computer*; *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Intelligent Systems*; *IEEE Computing in Science and Engineering*; *IEEE Transactions on Systems, Man, and Cybernetics*; *Computational Intelligence*; *Information and Management*; *Decision Support Systems*; *Journal of Systems and Software*; *Concurrency and Computation: Experience and Practice*. Prof. Zhuge is a Senior Member of IEEE and member of the ACM. He was among the top scholars (2000–2004, 2000–2005) in Systems and Software Engineering area according to the assessment report published in *Journal of Systems and Software.*