



An inexact model matching approach and its applications

Hai Zhuge *

Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, 100080 Beijing, PR China

Received 13 July 2001; received in revised form 3 January 2002; accepted 26 February 2002

Abstract

Matching between two model specifications is the key step of the repository-based model reuse. From the approximate specification point of view, this paper presents a quantified inexact matching theory for flexible model retrieval from large-scale model repositories. The theory specifies a model repository as two levels: a model level based on a multi-valued model specialization relationship and a fundamental function level based on a function specialization relationship. The matching degree between two models depends on their matching functions. The matching degree between two functions on a function specialization graph depends on the function-distance between them. A set of model specialization rules enables a new matching to be derived from the existing matchings. Embedded in an SQL-like command, the theory has been applied to a large-scale mathematical software model repository system. Users can use the command to retrieve the required models with an inexact query condition. Applications show that the approach is useful and tractable.

© 2002 Elsevier Inc. All rights reserved.

Keywords: Inexact retrieval; Model reuse; Model repository; Matching; Signature

1. Introduction

Model reuse techniques (Batory and O'Malley, 1992; Felice, 1993; Neighbors, 1984; Paul and Prakash, 1994; Rajlich and Silva, 1996; Ramamoorthy et al., 1998; Wohlin and Runeson, 1994) play an important role in promoting the productivity of the development of information systems (e.g., DSS and MIS). Model repository (Banker et al., 1993; Maarek et al., 1991; Yau and Tsai, 1987) is a mechanism to realize model reuse by retrieving the required model(s) from a set of well-defined models stored in a repository mechanism. Retrieval approaches based on exact matching can only get the exactly required models. They are not suitable for the following cases: (1) the exactly required model does not exist in model repository, while models with similar characteristics in model repository can also be reused (or reused after slightly modification); (2) domain users do not know (or cannot express) exactly what they need, just hope to present an inexact query requirement and simply choose the required models in a set of similar models returned by the retrieval mechanism; and (3)

users want to get acquaintance with the content of the large-scale model repository through inexact queries.

Retrieval approach based on inexact model matching is to find the model(s) approximate to users' query requirement. It can enhance the reusability of model repository and enables users to use and manage model repository conveniently and flexibly. Inexact model matching concerns three related aspects: (1) the description of the repository models, (2) the establishment of the inexact relationship between repository models, and (3) the quantification of the inexactness.

Pure formal techniques have a good mathematical basis for specification, verification and validation (Diaz and Groz, 1993; Ehrig and Mahr, 1985; ISO, 1989; Mili et al., 1997; Winstanley and Bustard, 1991). They emphasised on the completeness, correctness, preciseness and consistency. While in practice, they are not easy to be handled by domain users or ordinary programmers.

The signature-based approaches focus on the type information about the interface of a software component (e.g., Zaremski and Wing, 1995). A signature matching can be used for model retrieval and it can be further relaxed by using a specialization (or generalization) of signatures (Zhuge, 1998; Zhuge, 2000). But, the indirect specialization (caused by transitivity) and

* Fax: +86-106-256-7724.

E-mail address: zhuge@ict.ac.cn (H. Zhuge).

impure specialization (e.g., partial specialization), which are neglected in previous works, often exist in real applications besides the pure specialization relationship. The model matching also needs to be quantified so as to meet the need of domain users' inexact retrieval.

Informal approaches include the feature-based, the taxonomy-based, and the knowledge-based approaches (Boisvert et al., 1985; Devanbu, 1991; Ostertag et al., 1992; Zhuge, 1998). These approaches use outward characteristics of model to distinguish one model from the others. A good informal model specification should be informative and a proper balance between comprehensibility and preciseness under domain restraint. The model specification with the input and output information can be easily accepted by traditional programmers and domain users. It enables users to use models (especially those designed in non-OO paradigm) in their familiar way: calling a model with a set of parameter values, like the IMSL mathematical computing models and component object model (COM) interface specification.

From the standpoint of approximate specification, this paper presents a model retrieval approach based on a quantified similar signature matching. The application background is a large-scale mathematical model repository. The proposed approach enables two model specifications to be matched in a similarity degree within [0,1]. We established two specification levels for model repository: a fundamental function level and a higher model level. By defining a multi-valued model specialization relationship and a function specialization relationship at these two levels, we characterise the similarity between repository models. The matching degree between two models is based on their matching functions. The matching degree between two functions is measured by the longest function-distance between them on a function specialization graph (FSG). Furthermore we propose a set of rules that reflects the logical relationship between different values of the model specialization. These rules enable a new matching between two models to be derived from the existing matchings.

2. Model specialization

A model can be regarded as a set of functions from the viewpoint of functional programming, and it is also true in many applications. We herein define the model specialization based on function specialization.

2.1. Function specialization

A function can be characterised by a name and a type, denoted as *FunctionName: FunctionType*. *FunctionType* is a mapping between two types: $\sigma \rightarrow \tau$. σ and τ are called input type and output type respectively, and

both of them can be either basic type or constructed type. For example, the function carrying out the real (R) matrix (M) addition (Add) is named as $MAddR$, its type can be represented as: $MR \times MR \rightarrow MR$, where “ MR ” denotes the type of real matrix.

Definition 1. Let $f: \sigma \rightarrow \tau$ and $f': \sigma' \rightarrow \tau'$ be two functions, if we have: (1) $\sigma' \subseteq \sigma$, $\tau' \subseteq \tau$, and for any $\xi \in \sigma$, $\xi' \in \sigma'$, if $\xi = \xi'$, then $f(\xi) = f'(\xi')$; and, (2) the axioms of f' imply the axioms of f , then f' is a function specialization of f , denoted as $f \xrightarrow{F} f'$ (or in simple $f \rightarrow f'$). Otherwise, f' is not a specialization of f , denoted as $f \not\xrightarrow{F} f'$.

Function specialization is a relaxation of function equivalence, and can be used to establish a kind of similar relationship between functions. Such a similar relationship provides a basis for flexible function matching.

Characteristic 1. Function specialization satisfies: (1) $f \xrightarrow{F} f$; and, (2) $f \xrightarrow{F} f'$, $f' \xrightarrow{F} f'' \Rightarrow f \xrightarrow{F} f''$.

2.2. Model specialization

We herein use signature to describe the models from the standpoint of applicability. A model is represented as: $\text{Model Name} = \{f_1: \sigma_1 \rightarrow \tau_1, \dots, f_n: \sigma_n \rightarrow \tau_n\}$, within which there does not exist function duplication. Assertions for guaranteeing the integration of a model are regarded as functions. In the following discussion, the capital letter M , M' or M , are used to denote the model signature.

Definition 2. Model specialization is a behaviour compatible similar relationship between two models, the similarity degree can be characterised by multiple values: (1) if there exist a one-to-one and onto mapping from M into M' , and for every $f \in M$, there exists an $f' \in M'$ such that $f \xrightarrow{F} f'$, then M' is called an identical-specialization from M , denoted as $M \xrightarrow{I} M'$; (2) if for every $f' \in M'$, there exists an $f \in M$ such that $f \xrightarrow{F} f'$, then M' is called a partial-specialization from M , denoted as $M \xrightarrow{P} M'$; (3) if for every $f \in M$, there exists an $f' \in M'$ such that $f \xrightarrow{F} f'$, then M' is called an extension-specialization from M , denoted as $M \xrightarrow{E} M'$; (4) if there exists an $f \in M$ and an $f' \in M'$, such that $f \xrightarrow{F} f'$, then M' is called a revision-specialization from M , denoted as $M \xrightarrow{R} M'$; and, (5) if there does not exist an $f \in M$ and an $f' \in M'$, such that $f \xrightarrow{F} f'$, then M' is called a non-specialization from M , denoted as $M \not\xrightarrow{F} M'$.

The multi-valued model specialization establishes a kind of multi-valued similar relationship between models, which can be used to realize flexible model reuse.

Among these values of the model specialization, the identical-specialization is the strongest, and the non-specialization is the weakest. Both the partial-specialization and the extension-specialization are weaker than the identical-specialization and stronger than the revision-specialization. The revision specialization is stronger than the non-specialization. A graphical explanation of the four model specialization values and the specialization lattice are shown in Fig. 1.

2.3. Specialization graphs

A model repository specification consists of a model level and a function level as shown in Fig. 2. The function level consists of the fundamental and small-granularity functions. The model level consists of the application-oriented and large-granularity functions (models), which can be composed by the fundamental functions.

A FSG is a graph in which nodes are functions, and arcs are function specialization relationships. The function level of a model repository consists of several sub-graphs of the FSG (FSG_k , see the dashed circles at the function level in Fig. 2). Functions within the same FSG_k are related by the function specialization relationship. Two different FSGs are related by the non-specialization relationship. The basic management operations on the FSG include operations for appending and deleting function nodes. To append a new function to an FSG needs to append the function node and to establish the specialization relationship with its specialization predecessors and successors. To delete a function node needs to delete the node and all the specialization arrows related to it.

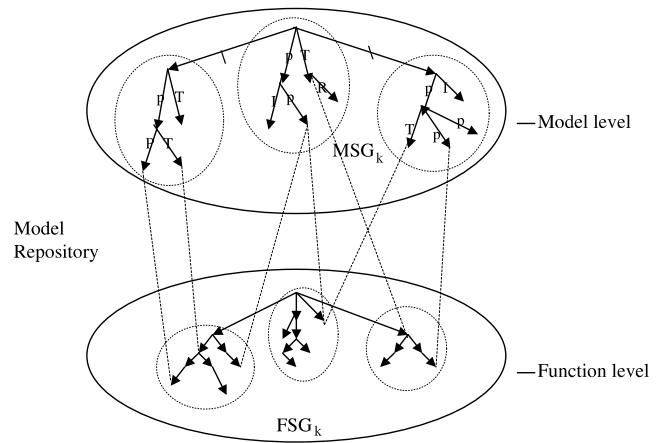


Fig. 2. Graphical explanation of MSG and FSG in a model repository.

A model specialization graph (MSG) is a graph in which nodes are models, and arcs are the multi-valued model specialization relationship. The model level of the model repository consists of several sub-graphs of MSG (MSG_k , see the dashed circles at the model level in Fig. 2). Models in the same sub-graph are related by the multi-valued model specialization relationship. Two different MSGs are related by the non-specialization relationship. A model in a MSG can be the composition of several functions in an FSG_k . The dashed lines between the two levels in Fig. 2 represent the composition relationship. The basic operations on the MSG include operations for appending and deleting nodes. To append a new model to a MSG needs to append the node and to establish the multi-valued model specialization relationship with its specialization predecessors and successors. To delete a model node

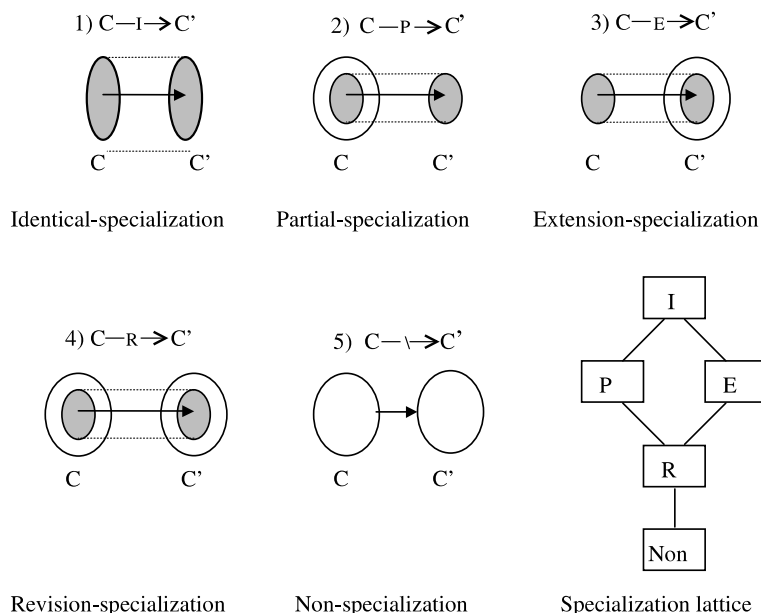


Fig. 1. Graphical explanation of the multi-valued model specialization.

needs to delete the node and all the specialization arrows related to it.

More large-granularity model specification levels can be layered upon the basic two-level specialization hierarchies, where every high-level model is the composition of the low-level models. The matching between two high-level models can be similarly defined on the basis of the matching between their direct low-level models.

3. Inexact model matching

3.1. Function distance and function matching

Let f_Q and f_M be two nodes in the FSG (f_Q can also be a query specification). There are three possible relationships between f_Q and f_M on the FSG: (1) f_Q is more special than f_M , (2) f_Q is the same as f_M and (3) f_Q is more general than f_M (i.e., f_M is more special than f_Q). They are denoted by $s = -1$, $s = 0$, and $s = 1$ respectively.

A function-distance $d(f_Q, f_M)$ is defined as the number of specialization arrows on the longest (i.e., the worst case) specialization path from f_Q to f_M on the FSG and satisfies:

$$d(f_Q, f_M) = \begin{cases} 0 & \text{if } f_Q \text{ is the same as } f_M; \\ 1 & \text{if } f_M \text{ is a direct specialization successor of } f_Q; \\ d(f_Q, f_k) + d(f_k, f_M) & \text{if } f_k \text{ is on the path from } f_Q \text{ to } f_M; \\ -d(f_M, f_Q) & \text{if } f_Q \text{ is the specialization successor of } f_M; \\ +\infty & \text{if there does not exist a path from } f_Q \text{ to } f_M \end{cases} \quad (1)$$

The function-distance reflects a comparative function specialization degree from one function node to another on the same function specialization path. The longer function distance between two functions provides more candidates similar to them. To append or to delete a function node may change the length of the longest specialization path between two functions, so the function-distance needs to be re-computed if the longest path has been changed.

The function matching between two functions f_Q and f_M denoted as $\text{Match}_f(f_Q, f_M, s)$, can be defined by the function-distance between them as follows:

$$\text{Match}_f(f_Q, f_M, s) = \gamma(1/(|d(f_Q, f_M)| + 1)) \quad (2)$$

where $s \in \{0, 1, -1\}$ is a variable for controlling the search direction. ‘ $s = 0$ ’ denotes the ‘exact matching’, ‘ $s = 1$ ’ denotes ‘ f_Q is more general than f_M ’, and ‘ $s = -1$ ’ denotes ‘ f_Q is more special than f_M ’. γ is a mapping, $\gamma: [0, 1] \rightarrow [0, 1]$, such that: (1) $\gamma(0) = 0$; (2) $\gamma(1) = 1$; and, (3) for any $a, b \in (0, 1)$, if $a \geq b$, then $\gamma(a) \geq \gamma(b)$. $\gamma(x) = x$ is a simple form of γ . In terms of the transitivity

characteristic of the function specialization, we have the following characteristic.

Characteristic 2. Let f_k be a function on the specialization path from f_Q to f_M then we have: $\text{Match}_f(f_Q, f_M, s) \leq \text{Match}_f(f_Q, f_k, s)$, and $\text{Match}_f(f_Q, f_M, s) \leq \text{Match}_f(f_k, f_M, s)$.

3.2. Computing similarity between two models

Let $|M_Q|$ and $|M_M|$ be the number of functions included in M_Q and M_M respectively. MF is a one-to-one and onto match-making function that enables any function f_Q of M_Q to match either a function of M_c (i.e., $MF(f_Q)$) or a null value, such that $d(f_Q, MF(f_Q))$ is the biggest one of all candidates. The similarity degree (sd) between M_Q and M_M is defined as follows:

$$sd(M_Q, M_M, s) = 2 \times \frac{\sum_{f_Q \in M_Q, f_M = MF(f_Q)} \text{Match}_f(f_Q, f_M, s)}{(|M_Q| + |M_M|)} \quad (3)$$

where $s = 0, 1$ and -1 denote the ‘exact matching’, ‘ M_Q is more general than M_M ’, and ‘ M_Q is more special than M_M ’ respectively. Since $\text{Match}_f(f_Q, f_M, s) \in [0, 1]$, we have $sd(M_Q, M_M, s) \in [0, 1]$. If both M_Q and M_M have

only one function (i.e., $M_Q = \{f_Q\}$ and $M_M = \{f_M\}$), then $sd(M_Q, M_M, s) = \text{Match}_f(f_Q, f_M, s)$.

The model matching between a model (or a query) M_Q and the model M_M in model repository is defined as follows:

$$\text{Match}_M(M_Q, M_M, sd, s) = \begin{cases} \text{true,} & \text{if } sd \geq \beta; \\ \text{false,} & \text{otherwise.} \end{cases} \quad (4)$$

where $sd \geq \beta$ means that the similarity degree (sd) between M_Q and M_M is equal to or bigger than a user’s expected value $\beta, \beta \in [0, 1]$.

3.3. Function distance matrix and model similarity matrix

A function-distance matrix FDM is used to record the initial values of the function-distances at the function level of a model repository, denoted as $FDM = (fd_{ij})_{n \times n}$, where n is the number of the total functions in model repository, fd_{ij} represents the function distance between function f_i and function f_j i.e., $d(f_i, f_j)$, and satisfies: $fd_{ii} = 0$; $fd_{ij} = -fd_{ji}$; and, $fd_{ij} \in (-\infty, \infty)$.

The model similarity degrees are recorded in a model specialization matrix (MSM), denoted as $MSM = (sd_{ij})_{p \times p}$, where p denotes the number of total models in model repository, sd_{ij} represents the similarity degree between M_i and M_j , i.e., $sd(M_i, M_j, s)$, and satisfies: $sd_{ii} = 1$; $sd_{ij} = sd_{ji}$; and, $sd_{ij} \in [0, 1]$.

The FDM and MSM are used to avoid computing a matching that has been computed. For the purpose of convenient retrieval and management, RDBMS-based relational tables can be used for managing the FDM and the MSM.

4. Matching derivation

Matching derivation is to get a new matching directly from the existing matchings rather than to compute it from scratch. Before discussing this issue, we first examine the logical relationship between the values of model specialization.

4.1. Rules for model specialization and meaningful connections

Logical relationship among the values of the model specialization can be represented by rules shown in Table 1, where ‘,’ denote logical AND, ‘ $X \rightarrow Y$ ’ means ‘if X is true, then Y is true’. Except for the revision-specialization, the other values of the model specialization have the transitivity characteristic as Rule 1 in Table 1. Rules 2–6 show the implication relationship between two different values of the model specialization. The transitivity and implication characteristics enable different values of the model specialization to be connected for derivation. Rules 7–12 shown in Table 1 represent such a derivation.

Proof of Rule 1. Let $\gamma = 'I'$, $M_1 \text{---} I \text{---} M_2$ and $M_2 \text{---} I \text{---} M_3$, according to Definition 2, there exist a one-to-one and

onto mapping θ and θ' , such that: (1) for every $f \in M_1$ there exist $f' \in M_2$, $f' = \theta(f)$ and $f \text{---} F \text{---} f'$; (2) for every $f' \in M_2$ there exists $f'' \in M_3$, $f'' = \theta'(f')$ and $f' \text{---} F \text{---} f''$. According to the transitivity of ‘=’ and ‘ $\text{---} F \text{---}$ ’ (Characteristic 1), we have: for any $f \in M_1$ there exist $f'' \in M_3$, $f'' = \theta'(\theta(f))$, and $f \text{---} F \text{---} f''$. Hence $M_1 \text{---} I \text{---} M_3$, i.e., Rule 1 holds when $\gamma = 'I'$. Similarly, we can prove Rule 1 holds when $\gamma = 'P'$ and $\gamma = 'E'$. Therefore Rule 1 holds.

Rules 2–6 can be easily proved by Definition 2. Rules 1–6 form the basic set of the specialization rules, more rules can be derived from this set. We herein present the proof of Rule 7. Rules 8–12 can be similarly proved. \square

Proof of Rule 7. Since $M_1 \text{---} I \text{---} M_2 \Rightarrow M_1 \text{---} P \text{---} M_2$ (by Rule 2), we have: $M_1 \text{---} I \text{---} M_2$, $M_2 \text{---} P \text{---} M_3 \Rightarrow M_1 \text{---} P \text{---} M_2$, $M_2 \text{---} P \text{---} M_3 \Rightarrow M_1 \text{---} P \text{---} M_3$ (by Rule 1). \square

The proposed rules can be used for: (1) establishing a rule reasoning mechanism for heuristic model-retrieval mechanism; (2) checking the consistency and correctness of the specialization relationship and the similarity between the models in model repository, e.g., $sd(M_1, M_3, s) \leq sd(M_1, M_2, s)$ can be a checking condition corresponding to Rule 10; and, (3) deriving a new matching from the existing matchings.

4.2. Matching derivation

Now we discuss how to get a new matching by deriving from the existing matchings. We use notation ‘ \cdot ’ to denote the connection of two existing matchings, e.g., $\text{Match}_M(M_1, M_2, sd_3 \geq \beta_3, s) \cdot \text{Match}_M(M_2, M_3, sd_3 \geq \beta_3, s)$. Since the revision-specialization does not have the transitivity characteristic, a matching connection may not be meaningful. The following definition defines a meaningful connection.

Definition 3. Let $\gamma, \varphi \in \{I, P, E, \setminus\}$, $M_1 \text{---} \gamma \text{---} M_2$, and $M_2 \text{---} \varphi \text{---} M_3$. If there exist $\theta \in \{I, P, E, \setminus\}$ and a rule: $M_1 \text{---} \gamma \text{---} M_2, M_2 \text{---} \varphi \text{---} M_3 \Rightarrow M_1 \text{---} \theta \text{---} M_3$, then the connection of $\text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s)$ and $\text{Match}_M(M_2, M_3, sd_2 \geq \beta_2)$ is meaningful, otherwise is not meaningful.

A new matching can be derived from a meaningful connection of the existing two matchings as follows:

$$\text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \cdot \text{Match}_M(M_2, M_3, sd_2 \geq \beta_2, s) \\ \Rightarrow \text{Match}_M(M_1, M_3, sd_3 \geq \beta_3, s) \quad (5)$$

where $\beta_3 = v_1(\beta_1, \beta_2)$ such that $\beta_3 \leq \beta_i$ ($i = 1, 2$) and $\beta_3 \in [0, 1]$ hold. In practice, the minimum function, ‘*Min*’ can be a simple choice of function v .

The derivation can be extended to the connection of more than two matchings. Assume we have: $M_1 \text{---} \gamma_1 \text{---} M_2, M_2 \text{---} \gamma_2 \text{---} M_3, \dots, M_{n-1} \text{---} \gamma_{n-1} \text{---} M_n \Rightarrow M_1 \text{---} \gamma_n \text{---} M_n$. The

Table 1
Rules for model specialization

Rule No.	Rules
Rule 1	$M_1 \text{---} \gamma \text{---} M_2, M_2 \text{---} \gamma \text{---} M_3 \Rightarrow M_1 \text{---} \gamma \text{---} M_3$, where $\gamma \in \{I, P, E\}$
Rule 2	$M_1 \text{---} I \text{---} M_2 \Rightarrow M_1 \text{---} P \text{---} M_2$
Rule 3	$M_1 \text{---} I \text{---} M_2 \Rightarrow M_1 \text{---} E \text{---} M_2$
Rule 4	$M_1 \text{---} I \text{---} M_2 \Rightarrow M_1 \text{---} R \text{---} M_2$
Rule 5	$M_1 \text{---} E \text{---} M_2 \Rightarrow M_1 \text{---} R \text{---} M_2$
Rule 6	$M_1 \text{---} P \text{---} M_2 \Rightarrow M_1 \text{---} R \text{---} M_2$
Rule 7	$M_1 \text{---} I \text{---} M_2, M_2 \text{---} P \text{---} M_3 \Rightarrow M_1 \text{---} P \text{---} M_3$
Rule 8	$M_1 \text{---} I \text{---} M_2, M_2 \text{---} E \text{---} M_3 \Rightarrow M_1 \text{---} E \text{---} M_3$
Rule 9	$M_1 \text{---} I \text{---} M_2, M_2 \text{---} R \text{---} M_3 \Rightarrow M_1 \text{---} R \text{---} M_3$
Rule 10	$M_1 \text{---} P \text{---} M_2, M_2 \text{---} R \text{---} M_3 \Rightarrow M_1 \text{---} R \text{---} M_3$
Rule 11	$M_1 \text{---} E \text{---} M_2, M_2 \text{---} R \text{---} M_3 \Rightarrow M_1 \text{---} R \text{---} M_3$
Rule 12	$M_1 \text{---} R \text{---} M_2, M_2 \text{---} \setminus \text{---} M_3 \Rightarrow M_1 \text{---} \setminus \text{---} M_3$

matching between M_1 and M_n can be got by the following derivation:

$$\begin{aligned} & \text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \\ & \cdot \text{Match}_M(M_2, M_3, sd_2 \geq \beta_2, s) \\ & \cdots \text{Match}_M(M_{n-1}, M_n, sd_{n-1} \geq \beta_{n-1}, s) \\ & \Rightarrow \text{Match}_M(M_1, M_n, sd_n \geq \beta_n, s) \end{aligned} \quad (6)$$

where s and $\beta_n = v(\beta_1, \beta_2, \dots, \beta_{n-1})$, such that $\beta_n \leq \beta_i$ ($i = 1, 2, \dots, n-1$) and $\beta_n \in [0, 1]$ hold. Since the smallest similar part can always be propagated, the minimum function, ‘*Min*’ is also a choice of function v .

Now we concern the connection of two matchings that do not have a common model, e.g., $\text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \cdot \text{Match}_M(M_3, M_4, sd_2 \geq \beta_2, s)$. In this case, we need to first examine whether such a connection is meaningful or not.

Definition 4. Let $\gamma, \varphi \in \{I, P, E, \setminus\}$, $M_1 \xrightarrow{\gamma} M_2$, and $M_3 \xrightarrow{\varphi} M_4$. If there exist $\mu, \theta \in \{I, P, E, \setminus\}$, such that $M_2 \xrightarrow{\mu} M_3$ and $M_1 \xrightarrow{\theta} M_4$, then $\text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \cdot \text{Match}_M(M_3, M_4, sd_2 \geq \beta_2, s)$ is meaningful, otherwise is not meaningful.

If $\text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \cdot \text{Match}_M(M_3, M_4, sd_2 \geq \beta_2, s)$ is meaningful, a new matching can be formed by the following derivation:

$$\begin{aligned} & \text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \cdot \text{Match}_M(M_3, M_4, sd_2 \geq \beta_2, s) \\ & \Rightarrow \text{Match}_M(M_1, M_2, sd_1 \geq \beta_1, s) \\ & \cdot \text{Match}_M(M_2, M_3, sd_3 \geq \beta_3, s) \\ & \cdot \text{Match}_M(M_3, M_4, sd_2 \geq \beta_2, s) \\ & \Rightarrow \text{Match}_M(M_1, M_4, sd_4 \geq \beta_4, s), \end{aligned} \quad (7)$$

where $\beta_4 = v(\beta_1, \beta_2, \beta_3)$, such that $\beta_4 \leq \beta_i$ ($i = 1, 2, 3$) and $\beta_4 \in [0, 1]$ hold. Since the smallest similar portion can always be propagated, the minimum function, ‘*Min*’ is also a choice of function v .

5. Inexact model retrieval

5.1. Query command

Inexact model retrieval can be realized by an SQL-like command, which has the similar semantic as the conventional SQL. The syntax of the command is described as follows:

```
<SelectCommand> ::= SELECT <Quantifier> [M]
[FROM MR | (<SelectCommand>)]
[WHERE <Condition>] [GROUP INTO M]
<Quantifier> ::= All | Unique | Direct | AllDirect |
UniqueDirect
<Condition> ::= M1 - I - > M2 | M1 - P - > M2 | M1 - E - >
M2 | M1 - R - > M2 | M1 - \ - > M2 | LIKE M |
```

Table 2
Refinement of query conditions

Condition	User's familiarity degree about target (%)	Classification
Like M	10%-30%	Inexact
$Sd(M, *, s) \geq \beta$	10-50	Inexact
$M = \text{String} \sim$	10-60	Inexact
$M > M'$ or $M < M'$	50-70	Inexact
$M \xrightarrow{R} M'$	50-80	Inexact
$M \xrightarrow{P} M'$	80-90	Inexact
$M \xrightarrow{E} M'$	80-90	Inexact
$M \xrightarrow{I} M'$	90-100	Inexact
$M = M'$	100	Exact

Top-Down
Refinement
↓

$$\begin{aligned} & M_1 < M_2 | M_1 > M_2 | M_1 = M_2 | MN = \langle \text{String} \rangle \sim | sd(M_1, \\ & *, s) \geq \beta | \\ & \langle \text{Condition} \rangle \text{ AND } \langle \text{Condition} \rangle | \langle \text{Condition} \rangle \\ & \text{OR } \langle \text{Condition} \rangle | (\langle \text{Condition} \rangle) \end{aligned}$$

In the above command, the brackets, $[\dots]$, mean the content by default. The $\langle \text{quantifier} \rangle$ expresses the restriction to the required target: ‘Direct’ refers to the direct specialization successor of a model; ‘Unique’ refers to the query requirement of finding a unique model among several candidates; ‘Direct’ refers to the query requirement of finding a direct successor/predecessor; ‘AllDirect’ refers to find all the direct successors/predecessors; and, ‘UniqueDirect’ refers to the query requirement of finding a unique direct successor/predecessor. ‘From MR’ means that from which repository the model could be retrieved. The ‘From SelectCommand’ subclause is an embed command for limiting the search scope. The model specialization relationship is represented in the condition part of the command. ‘MN’ denotes the model name. $MN = \langle \text{String} \rangle \sim$ means that ‘ $\langle \text{String} \rangle$ ’ is the initial part of the name string of M. If the model name is coded to reflect the model category, users can use the condition to limit the retrieval scope within a category, e.g., $\text{Select } M = MMul \sim$, where $MMul$ stands for a category of models. ‘ $M_1 < M_2$ ’ and ‘ $M_1 > M_2$ ’ mean that M_1 is more special and more general than M_2 respectively.

The embed command and the specialization hierarchy of a model repository support a refinement retrieval strategy. Users can retrieve the required models by using different conditions according to their familiarity degrees about the targets. A refinement order among the query conditions are shown in Table 2.

5.2. Process for model repository development

The proposed theory can either be incorporated into the conventional model repository development process or be used to establish a specification level upon the

existing conventional model repository to realize inexact model retrieval. The related development process consists of the following steps:

- (1) Specify the signature for every model and all the included functions;
- (2) Establish the FSG and a relational table for FDM;
- (3) Establish the MSG and a relational table for MSM;
- (4) Design the maintenance mechanism for FSG, FDM, MSG, and MSM; and,
- (5) Design the inexact retrieval mechanism.

Step 1 should be accomplished by the designers or the specialized users. Steps 2–3 can be accomplished by either the designers or assistant tools. Users can adjust the contents in the MSG and the FSG with the help of a management mechanism. Steps 4–5 should be accomplished by the designers.

6. Application

6.1. Overview

The proposed similar matching theory has been applied to a problem-oriented model repository system PROMBS (Zhuge, 2000). The system is developed for improving the existing FORTRAN-based mathematical software library (e.g., IMSL Math/Library) in three aspects: (1) provide an inexact query mechanism for users (especially beginners) in the scientific computing field to flexibly retrieve model resources (may be coded in different languages); (2) establish a kind of similar relationship in the model repository so as to increase the reusability of models; and, (3) provide a problem-oriented specification and problem-solving support mechanism.

The model repository of the PROMBS is separated into an upper specification level (which further consists of a feature level and a signature level) and a low implementation level. The model specialization relationships and the function specialization relationships are established at the signature level. The system also provides a problem-oriented high-level description language POL for users to describe the solutions (high-level programs) to the problems by using the specified repository models (Zhuge, 2000). A translator mechanism translates the high-level programs into 3GL (e.g., FORTRAN) programs. The specification level models are mapped into the implementation level models during the translation process.

6.2. Function specialization relaxation

Definition 1 is a general definition of the function specialization. In real application, the axiom expression

is not easy for ordinary developers (or users) to deal with, and the comparison between two axioms are time-consuming. In practice, different relaxation forms of the definition can be used in real applications. Any relaxation of the definition should keep its first condition and relax its second condition. One way to relax the second condition is to use the taxonomy-based approach to differentiate one function from another in domain applications.

In system PROMBS, the name of the specialized function is coded for identifying its behaviour category by extending the name string of the function to be specialized according to a set of naming rules. The specialization relationship between two functions can be judged by comparing their signatures and checking their names according to the naming rules. For example, the function for computing the positive (P) real (R) addition (Add) can be represented as $AddRP: PositiveReal \times PositiveReal \rightarrow PositiveReal$, the function for computing real addition can be represented as $AddR: Real \times Real \rightarrow Real$. Since $PositiveReal \subseteq Real$ and $AddR$ is a sub-string of $AddRP$, we have $AddR \xrightarrow{F} AddRP$. In PROMBS, classification information is also included in model names according to a set of domain-dependent naming rules like the classification scheme in (Boisvert et al., 1985). Generally, naming rules should enable a name to be understandable and short. Different application domains require different naming rules, e.g., the GUID of COM model (<http://www.microsoft.com/com>) is different from the GAMS (Boisvert et al., 1985).

A promising way to determine the function specialization relationship is to establish the function ontology mechanisms for application domains just as the WordNet (www.cogsci.princeton.edu/~wn). Assistant tools are needed to edit and manage the ontology. The ontology provides a reuse basis for the high-level applications. The Semantic Web (see <http://www.semanticweb.org>) provides the basis for representing and sharing functions across the Internet.

6.3. Heuristic rules

According to the relaxation of the model specialization, a set of heuristic rules can be formed and incorporated into the search mechanism. These rules are for: search navigation, specialization relationship judgement, and consistency checking among specialization relationships. For example, the following two heuristic rules are used for search navigation and for judging the function specialization relationship by name with a certainty degree (cd).

- (1) HeuristicRule1: IF CurrentNode $\xrightarrow{\gamma}$ Target THEN search the candidate node following the model specialization arc ' $\xrightarrow{\gamma}$ ' or the specialization arc that implies ' $\xrightarrow{\gamma}$ '.

- (2) HeuristicRule2: IF $name(f)$ is a sub-string of $name(f')$, THEN $f \xrightarrow{F} f'$ with $cd = 0.8$.

The multi-valued model specialization hierarchy can raise the efficiency of model retrieval. A key step of the model retrieval is to expand the proper successor(s) of the current visiting node. Assume the current visiting node be M_M and its successor set be $SCS(M_m)$. Each element in $SCS(M_M)$ is related to M_M by a value of the model specialization relationship. With the help of heuristic rules, the value of the model specialization between M_m and the candidate successor can be determined before expanding the successors. Hence, only a subset of $SCS(M_M)$ needs to be considered as the possible candidate expansion nodes, i.e., the search space can be narrowed.

6.4. Retrieval example

A set of models in the model repository of PROMBS concern matrix operations, like *MMulC* (complex matrix multiplication), *MMulAddR* (real matrix multiplication and addition), *MMulMinR* (real matrix multiplication and minus), *MMulAddRevR* (real matrix multiplication, addition and reverse), *MMulR* (real matrix multiplication), and *MMulS* (symmetric matrix multiplication). The related MSG and FSG are shown in Fig. 3, where the dashed lines denote some composition relationships, e.g., *MMulMinR* is the composition of *MMulR* and *MMinR*.

Users can use the query ‘SELECT ALL WHERE $sd(M_Q = \{MMulR : MR \times MR \rightarrow MR\}, *, 1) \geq 0.5$ ’ to retrieve all the models that are similar to M_Q with the similarity degree bigger than or equal to 0.5. The retrieval mechanism carries out the search following the specialization path, and returns the query result listed in Table 3. The values of the sd are computed by formula

Table 3
Returned models of a query

No.	Model name	sd
1	<i>MMulR</i>	1.00
2	<i>MMulRS</i>	0.50
3	<i>MMulAddR</i>	0.67
4	<i>MMulMinR</i>	0.67
5	<i>MMulAddRevR</i>	0.50

1, 2, 3, 4. We adopt the mapping $\gamma(x) = x$ when using formula (2).

If users’ query is ‘SELECT ALL WHERE $sd(M_Q = \{MMulR : MR \times MR \rightarrow MR\}, *, -1) \geq 0.5$ ’, the retrieval mechanism carries out the search following the opposite direction of the specialization path, and returns *MMulC* with $sd = 0.5$ and *MMulR* with $sd = 1$. The query ‘SELECT ALL WHERE $sd(M_Q = \{MMulR : MR \times MR \rightarrow MR\}, *, 0) = 1$ ’ is for exact matching, the return is *MMulR* with $sd = 1$. Through the inexact query, users can get the models similar to a reference model specification then determine to use a suitable one among the returned models.

7. Further applications

Since the proposed approach separates the specification level from the implementation level, the repository management mechanism can focus on managing the specification-level models. The implementation-level models can be many other kinds of entities such as source codes, binary codes, graphs, documents, etc. So the proposed approach is suitable for the other applications after making a concept mapping and adding domain specific restrictions due to the strategy of separating the specification level from the implementation level. This section discusses the application of the proposed approach to realize inexact software component retrieval and inexact knowledge retrieval.

7.1. Inexact software component retrieval

A small COM (see <http://www.microsoft.com/com/>) interface repository was built by using the presented repository building approach. To enable users to easily understand and to use components, the textual description and the class name are included into the component specification (this is an adaptation of the approach to suit a new application). Each component is specified by $ComponentName = \{Signature, Description, ClassName\}$. Table 4 lists four components of the repository.

The component name and the function name reflect the categories they belong to. Since all the functions

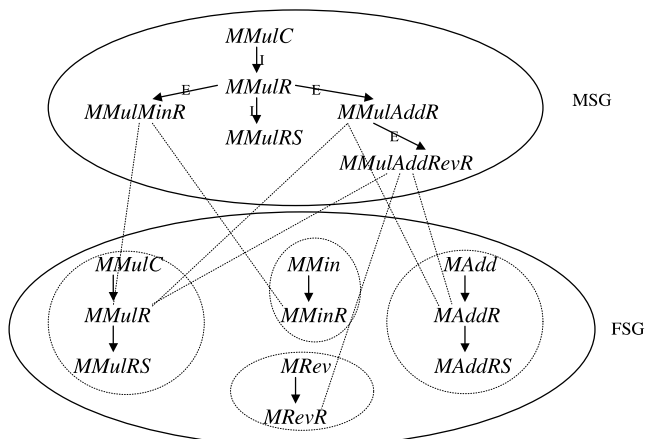


Fig. 3. An example of MSG and FSG.

Table 4
Component interface

Component name	Functions used
1 IUnKnow	{QueryInterface, AddRef, Release}
2 IOleWindow	{GetWindow, ContextSensitiveHelp}
3 IOlePlaceUIWindow	{QueryInterface, AddRef, Release, GetWindow, ContextSensitiveHelp, GetBorder, RequestBorderSpace, SetBorderSpace, SetActiveObject}
4 IOleInPlaceFrame	{QueryInterface, AddRef, Release, GetWindow, ContextSensitiveHelp, GetBorder, RequestBorderSpace, SetBorderSpace, SetActiveObject, InsertMenus, SetMenu, RemoveMenus, SetStatusText, EnableModeless, TranslateAccelerator}

(methods) in the COM components are uniquely designed, function specialization herein becomes function equivalence. But the FSG is still needed because new functions that are similar to the existing functions will be added. According to Definition 2, we have: (1) IOleInPlaceFrame is the extension specialization of the other three components, and (2) IOlePlaceUIWindow is the extension specialization of both IUnKnow and IOleWindow. Similar to the MSG, we can establish the component specialization graph as shown in Fig. 4.

Table 5 shows five inexact queries and the corresponding query results. The inexact retrieval approach can provide more query results to the user for reference. The previous COM repository retrieval mechanism is based on classification and keywords. The proposed approach provides a way to improve the existing repository mechanism by establishing specialization ties between components and providing the inexact retrieval mechanism. This example shows that the inexact model retrieval approach is suitable for inexact reuse of software components.

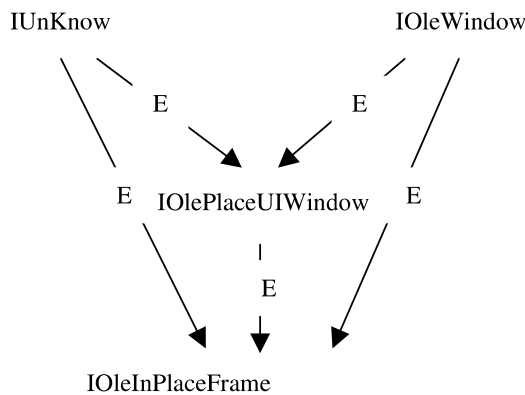


Fig. 4. Specialization hierarchy of four COM components.

Table 5
Inexact queries and query results

Queries	Return results
SELECT ALL WHERE $sd(M_Q = \{IOlePlaceUIWindow\}, *, 1) \geq 0.8$	IOlePlaceUIWindow
SELECT ALL WHERE $sd(M_Q = \{IOlePlaceUIWindow\}, *, 1) \geq 0.4$	IOleInPlaceFrame
SELECT ALL WHERE $sd(M_Q = \{IOlePlaceUIWindow\}, *, -1) \geq 0.4$	NO
SELECT ALL WHERE $sd(M_Q = \{IOlePlaceUIWindow\}, *, 1) \geq 0.25$	IunKnow
SELECT ALL WHERE $sd(M_Q = \{IOlePlaceUIWindow\}, *, 1) \geq 0.16$	IunKnow, IOleWindow

7.2. Inexact knowledge retrieval

The proposed approach can be applied to inexact reuse of general structured resources like the textual information, knowledge and services. A large-granularity resource consists of a set of related small-granularity resources. We herein discuss how to apply the proposed approach to realize inexact knowledge retrieval.

People usually use a set of related knowledge as an entire component (called Knowledge Closure) because the use of any portion of the Knowledge Closure sometimes does not make any sense. A Rule-Closure is one type of the Knowledge Closure. Each rule takes the following form: “Rule-name: IF condition THEN conclusion”. It can be simplified as: Rule-name: condition \rightarrow conclusion. If we regard the “condition” and the “conclusion” portion of a rule as the input type and the output type of a function respectively, a rule can be regarded as a function.

The mapping defined by the following five correspondences enables the proposed approach to be applicable to inexact knowledge retrieval.

- (1) Model \rightarrow Rule-Closure.
- (2) Function \rightarrow rule.
- (3) Function specialization \rightarrow rule specialization.
- (4) Model specialization \rightarrow Rule-Closure specialization.

Similar to the FSG and the MSG, we established the rule specialization graph and the Rule-Closure Specialization Graph according to the rule specialization relationship between rules and the Rule-Closure specialisation relationship between Rule-Closures. We have implemented the inexact knowledge retrieval in the Knowledge Grid project VEGA-KG (Zhuge, 2002a;

Zhughe, 2002b), where the rule specialization relationships are determined by the specialization relationship between the concepts used in the “condition” and the “conclusion” portions of rules. The demo of the knowledge retrieval is available at <http://kg.ict.ac.cn>.

8. Related works and discussion

8.1. Related works and comparisons

A feature-based description approach was used in AIRS (Ostertag et al., 1992). In the system, the similarity between two components is measured by a distance on the graph defined by the composition relationships and the closeness (i.e., a component is the modification of the other components) relational graph. The distance is computed by the weight of the arc on the graph. But the weight can be obtained only by domain analysis, experiment, or experience. It is not easy for conventional designers (or users) to determine the exact weight of the arc without the help of criteria, especially that between a newly added component and an existing component. Besides, assigning the value of the weight may vary with different determiners.

In knowledge-based software engineering context, description logics are used to describe software components (Devanbu and Jones, 1997). A knowledge-based software information system LaSSIE incorporates a large knowledge base, a semantic retrieval algorithm based on the formal inference, and a user interface incorporating a graphical browser and a natural language parser (Devanbu, 1991). Efforts were also made in developing software tools for supporting friendly formal specification interface. For example, a case-based reasoning system is developed to support ADT specification (Termsinsuwan et al., 1996). It can help to find a similar ADT and advise modifications. While the pre-specified ADT cannot cover the new domain applications, some required new specifications still need the “ADT-expert” to describe, and the modification of an ADT specification requires the user quite familiar with the ADT.

Component matching has been investigated at both the function level and the module level (Zaremski and Wing, 1995, 1997). Relaxation of the matching is based on a subtype specialization and generalization. But the impure specialization (or generalization) also plays an important role in practice. New specialization relationship can be derived from the transitivity of the existing specialization/generalization relationship. On the other hand, traditional signature expressions reflect only the type information about the interface, it cannot distinguish those components that have different behaviours but share the same interface type. An ideal component

repository should include not only the signature description but also the behaviours of each component.

Comparing with the previous approaches, the proposed approach has three main characteristics. First, the presented signature expression describes both the type information about the interface and the taxonomy information of a model. The type information about the interface of a model determines a set of models that have the same interface type, and the taxonomy information refines the specification within the set. As an implementation approach, we suggest to use name coding to realize the refinement. It is in line with the solutions in many current software systems. Second, we have enriched the semantic knowledge of the model repository by determining the objective relationship between the repository models: a multi-valued model specialization hierarchy and a function specialization hierarchy, which provide the basis for the extension of the matching relaxation. Only the essential domain knowledge is required for judging the function specialization relationship. Third, we provide an inexact model retrieval mechanism based on the inexact signature matching.

8.2. About the specialization

Three aspects need to be discussed about the model specialization and the function specialization. First, the generalization concept is close to the concept of specialization. The function generalisation is a reversal relationship of the function specialization. f can be called a generalization of f' if $f \text{---}F \text{---} f'$. Model generalization can be also regarded as a reversal relationship of the model specialization, defined by the following four items: (1) M is called an identical-generalization of M' if $M \text{---}I \text{---} M'$; (2) M is called a partial-generalization of M' if $M \text{---}E \text{---} M'$; (3) M is called an extension-generalization of M' if $M \text{---}P \text{---} M'$; and, (4) M is called a revision-generalization of M' if $M \text{---}R \text{---} M'$.

Second, function specialization can be defined in different point of views. For example, a mathematical function is a mapping $f: \sigma \rightarrow \tau$, where σ and τ are the domain and the co-domain of f . For $f: \sigma \rightarrow \tau$ and $f': \sigma' \rightarrow \tau'$, if $\sigma' \subseteq \sigma$, $\tau' \subseteq \tau$, and for any $\xi \in \sigma$, $\xi' \in \sigma'$, $\xi = \xi'$, we have $f(\xi) = f'(\xi')$, then f' is called the function specialization of f . From the viewpoint of programming, a function is a program performing a definite behaviour. We can say that f' is the function specialization of f , if the pre-condition of f' is weaker than the pre-condition of f and the post-condition of f' is stronger than or equivalent to the post-condition of f . For example in Eiffel language, the pre-condition and the post-condition are used to enhance the correctness of programs and to define the inheritance. The proposed definitions of model specialization and model matching

can be used to define different kinds of function specialization.

Third, the model specialization is different from the traditional inheritance in OOP context. First, the model specialization focuses on the similarity between two models. The behaviour compatibility between two models can be relaxed from compatible to incompatible. While the inheritance focuses on the incremental definition of code and method reuse. An inheritance descendent is usually not behaviour compatible with its ancestor. Second, the presented model specialization is defined at the signature level, which is completely separated from its implementation level. A model can be implemented in any language. While, the inheritance depends on the implementation of models. For example, different language has different inheritance mechanism. A model coded in one language (e.g., JAVA) cannot inherit from a model coded in another language (e.g., C++). It is unrealistic to require all the model programmers to use a common language. Hence traditional inheritance is not eligible for designing an open model repository (this is why COM does not support inheritance).

9. Summary

We have presented an inexact model-reuse approach based on a quantified inexact signature-matching theory. The main contribution of this work consists of two aspects. First, we proposed a theory of similar model matching. The theory can increase flexibility of model matching by determining the multi-valued model specialization relationship between repository models. A matching candidate can be either the exactly required model or the model that is approximate to the requirement. Second, we proposed an approach for implementing the theory and applied it to the large-scale mathematical software model repository in the scientific computing domain. The approach improves the flat repository and the exact retrieval mechanism of the domain without any modification on the existing models by establishing the two-level specialization hierarchies and realizing inexact model retrieval. The proposed approach is also suitable for the other related application fields.

Ongoing works are to apply the proposed approach to realize inexact reuse of web services and to realize inexact retrieval of Grid components on the Grid platform (Zhuge, 2002a; Zhuge, 2002b).

Acknowledgement

The research work was supported by the National Science Foundation (NSF) of China.

References

- Batory, D., O'Malley, S., 1992. The design and implementation of hierarchical software systems with reusable components. *ACM Trans. Soft. Eng. Methodol.* 1 (4), 355–398.
- Banker, R.D., Kauffman, R.J., Zweig, D., 1993. Repository evaluation of software reuse. *IEEE Trans. Soft. Eng.* 19 (4), 379–389.
- Boisvert, R.F., Howe, S.E., Kahaner, D.K., 1985. GAMS: A framework for the management of scientific software. *ACM Trans. Math. Soft.* 11 (4), 313–355.
- Devanbu, P. et al., 1991. LaSSIE a knowledge-based software information system. *Commun. ACM* 34 (5), 35–49.
- Devanbu, P., Jones, M.A., 1997. The use of description logics in KBSE systems. *ACM Trans. Soft. Eng. Methodol.* 6 (2), 141–172.
- Diaz, M., Groz, R. (Eds.), 1993. *Formal Description Techniques V.* North-Holland, Amsterdam, The Netherlands, pp. 65–80.
- Ehrig, H., Mahr, H., 1985. *Fundamentals of Algebraic Specification.* Part I. Springer-Verlag, Berlin.
- Felice, P.D., 1993. Reusability of mathematical software: a contribution. *IEEE Trans. Soft. Eng.* 19 (8), 835–843.
- ISO, 1989. Information processing systems—open systems interconnection—LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. ISO8807.
- Maarek, Y.S., Berry, D.M., Kaiser, G.E., 1991. An information retrieval approach for automatically constructing software libraries. *IEEE Trans. Soft. Eng.* 17 (8), 800–813.
- Mili, R., Mili, A., Mittermeir, R.T., 1997. Storing and retrieving software components: A refinement based system. *IEEE Trans. Soft. Eng.* 23 (7), 445–460.
- Neighbors, J.M., 1984. The Draco approach to constructing software from reusable components. *IEEE Trans. Soft. Eng.* SE-10 (5), 564–574.
- Ostertag, E. et al., 1992. Computing similarity in a reuse library system: An AI-based approach. *ACM Trans. Soft. Eng. Methodol.* 1 (3), 205–228.
- Paul, S., Prakash, A., 1994. A framework for source code search using program patterns. *IEEE Trans. Soft. Eng.* 20 (6), 463–475.
- Rajlich, V., Silva, J.H., 1996. Evolution and reuse of orthogonal architecture. *IEEE Trans. Soft. Eng.* 22 (2), 153–157.
- Ramamoorthy, C.V., Garg, V., Prakash, A., 1998. Support for reusability in Genesis. *IEEE Trans. Soft. Eng.* 14 (8), 1145–1153.
- Termsinsuwan, P., Cheng, Z., Shiratori, N., 1996. A new approach to ADT specification support based on reuse of similar ADT by the application of case-based reasoning. *Informat. Soft. Technol.* 38, 555–568.
- Winstanley, A.C., Bustard, D.W., 1991. EXPOSE: Animation tool for process-oriented formal descriptions. *Soft. Eng. J.* 6, 463–475.
- Wohlin, C., Runeson, P., 1994. Certification of software components. *IEEE Trans. Soft. Eng.* 20 (6), 494–499.
- Yau, S.S., Tsai, J.J., 1987. Knowledge representation of software component interconnection information for large-scale software modifications. *IEEE Trans. Soft. Eng.* 13 (3), 355–361.
- Zaremski, A.M., Wing, J.M., 1995. Signature matching: A tool for using software libraries. *ACM Trans. Soft. Eng. Methodol.* 4 (2), 146–170.
- Zaremski, A.M., Wing, J.M., 1997. Specification matching of software components. *ACM Trans. Soft. Eng. Methodol.* 6 (4), 333–369.
- Zhuge, H., 1998. Inheritance rules for flexible model retrieval. *Decision Support Syst.* 22 (4), 383–394.
- Zhuge, H., 2000. A problem-oriented and rule-based component repository. *J. Syst. Soft.* 50 (3), 201–208.
- Zhuge, H., 2002a. A knowledge grid model and platform for global knowledge sharing. *Expert Syst. Appl.* 22 (4), 313–320.
- Zhuge, H., 2002b. A knowledge flow model for Peer-to-Peer Team knowledge sharing and management. *Expert Syst. Appl.* 23 (1), 23–30.

Hai Zhuge is a full professor at the Institute of Computing Technology, Chinese Academy of Sciences. He was an associate professor at the Institute of Software, Chinese Academy of Sciences. He received the Ph.D. in computer science from Zhejiang University, China. He visited the National University of Singapore, the City University of Hong Kong, and the Tsinghua University several times as a senior visiting scholar or research fellow from 1994 to 2000. His current research interests include: knowledge management, grid and the semantic web, problem-oriented model base systems, component reuse, cognitive-based software process model, inter-operation model

for group decision, and web-based workflow model. He is now the principle investigator of the China Knowledge Grid project (Vega-KG) as well as three national grants. He is the author of one book and over 40 papers appeared mainly in leading international conferences and the following international journals: *IEEE Transactions on Systems, Man, and Cybernetics*; *Information and Management*; *Decision Support Systems*; *Journal of Systems and Software*; *International Journal of Cooperative Information Systems*; *Expert Systems with Applications*, *Information and Software Technology*; and, *Knowledge-based Systems*.