Information and Software Technology 44 (2002) 445–450

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

# A process matching approach for flexible workflow process reuse

Hai Zhuge*

*Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Knowledge Grid Group, Beijing 100080, People's Republic of China*

Received 18 September 2001; revised 27 December 2001; accepted 6 February 2002

**Abstract**

Matching between two workflow processes is the key step of workflow process reuse. This paper presents an inexact matching approach for flexible workflow process reuse. A multi-valued process specialization relationship is defined based on the definition of activity specialization and the characteristics of workflow process. The matching degree between two workflow processes is determined by the matching degrees of their corresponding sub-processes or activities. The matching degree between two activities is determined by the activity-distance between them in the activity-ontology repository. A set of process specialization rules enables a new process matching to be derived from the existing matchings. Users are provided with an SQL-like command to retrieve the required processes in an inexact query condition from the workflow-process-ontology repository. 

*Keywords*: Process reuse; Process matching; Ontology; Workflow

## 1. Introduction

A workflow is the computerized facilitation or automation of a business process, in whole or part [6–8,11,12]. A workflow process can be abstracted as a network with activity nodes and flows (i.e. transitions). A domain business process can be modelled through the execution of the network, and thus can be controlled and managed through incorporating the domain business into the execution process of the network. The benefits of the workflow-based applications, like flexibility, integration and reusability, are discussed [3].

Process reuse is a way to promote the efficiency and quality of workflow process design, just as the software component reuse techniques [1,4,5]. A workflow-process-ontology repository is a mechanism to store and manage the well-defined workflow processes. Workflow process reuse can be realized through retrieving the required process from the repository like the component repository [2,4,9].

The workflow process retrieval can be exact or inexact. The exact retrieval is to get the exactly required processes, while the inexact retrieval is to find the processes approximate to the users' requirement. The inexact retrieval is useful in the following two cases: (1) the exactly required process does not exist in the repository, while the similar processes in the repository are also usable; and (2) users have a vague requirement, and hope to present an inexact query then choose the required processes among a set of similar processes returned by the retrieval mechanism. The inexact retrieval can enhance the reusability of the existing repository processes.

The relaxation of software component matching was investigated based on the subtype specialization and the signature specification [9]. But, the impure specialization widely exists in real applications besides the subtype specialization. New specialization relationship could be further derived from the existing specialization relationships as discussed in Ref. [10].

This paper presents an inexact process matching approach that enables two processes to be matched with a similarity degree of [0,1]. We characterize the similarity between repository processes through defining a multi-valued workflow process specialization relationship and the activity specialization relationship. The matching degree between two workflow processes is determined by the matching degrees of their corresponding sub-processes or activities. The matching degree between two activities is measured by the longest activity-distance between them on an activity specialization graph. Furthermore, we present a set of rules that reflects the logical relationship among different values of the process specialization. These rules enable a new matching between two workflow processes to be derived from the existing process matchings. We

* Fax: +86-1062567724.
*E-mail address:* zhuge@ict.ac.cn (H. Zhuge).

0950-5849/02/$ - see front matter 
PII: S0950-5849(02)00022-8

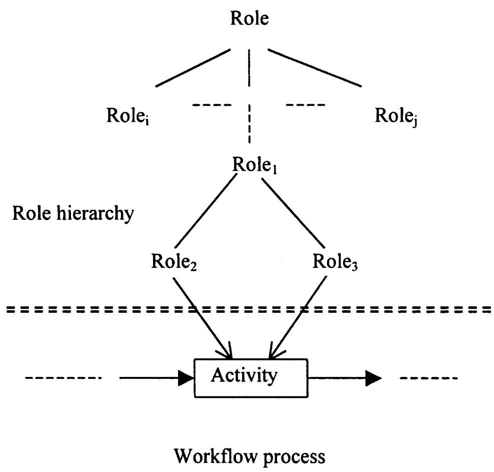


Fig. 1. Role hierarchy and the related activity in workflow process.

adopt the SQL-like command as the user interface to retrieve the required process in an inexact condition from the workflow-process-ontology repository.

## 2. Process specialization

### 2.1. Activity, role and activity specialization

An activity concerns three basic elements in the build time of a workflow: function, restrictions and participated roles, denoted as $A$: $(f{:}\sigma \rightarrow \tau, RES, ROLES)$, where $\sigma$ and $\tau$ are the input type and output type of function $f$, $RES$ is a restriction set, and $ROLES$ is a role set. A role can participate several activities, and an activity also enables several activities to participate. The roles participate an activity can be a hierarchy as shown in Fig. 1, where the high level role (i.e. super role) can play the role of the linked low level role(s) (called sub-role(s)). The role(s) participate(s) an activity is a part of the role hierarchy of the whole workflow process.

According to the specialization relationship between the elements of an activity, we define the activity specialization relationship as follows, where the function specialization relationship has been defined in Ref. [10].

**Definition 1.** Let $A$: $(f{:}\sigma \rightarrow \tau, RES, ROLES)$ and $A'$: $(f'{:}\sigma' \rightarrow \tau', RES', ROLES')$ be two activities, we say $A'$ is an activity specialization of $A$, denoted as $A \rightarrow A'$, if the following three items hold: (1) $f'$ is the specialization of $f$; (2) $RES'$ is equivalent to or stronger than $RES$; and (3) for any $r' \in ROLES'$ either we have $r' \in ROLES$ or there exists a $r \in ROLES$, $r$ is a super role of $r'$ (i.e. $r$ can play the role of $r'$).

Activity specialization is a relaxation of the activity equivalence. It provides a basis for flexible activity matching. According to the transitivity of the conditions of the activity specialization, we have the following characteristic.

*Characteristic 1*: Activity specialization satisfies: (1) $A \rightarrow A$; and (2) $A \rightarrow A'$, $A' \rightarrow A'' \Rightarrow A \rightarrow A''$.

### 2.2. Workflow process description and specialization

A workflow process (type) consists of a set of activities $AS = \{A_1, \ldots, A_n\}$, a set of connection conditions $CON = \{\phi$, sequential, and-join, and-split, or-join, or-split$\}$ (their semantics are available at Ref. [7]), and a set of connection operations $COP = \{PreCon, PostCon, Pre, Post\}$, represented as $ProcessName = \langle AS, CON, COP \rangle$, and for any $A_i \in AS$, its pre-condition $PreCon(A_i) \in CON$, its post-condition $PostCon(A_i) \in CON$, its predecessor $Pre(A_i) \in AS$, and its successor $Post(A_i) \in AS$. We call $A_i$ the initial activity and the termination activity, respectively, in case of $Pre(A_i) = \phi$ and $Post(A_i) = \phi$.

**Definition 2.** A workflow process P′ is called an identical-specialization from another workflow process P, denoted as P–I → P′, if there exists an isomorphism from P into P′ such that for every $A \in$ P, there exists a corresponding $A'$ of P′ and satisfies: (1) $A \rightarrow A'$; (2) $PreCon(A) = PreCon(A')$; (3) $PostCon(A) = PostCon(A')$; (4) $Pre(A) \rightarrow Pre(A')$; (5) $Post(A) \rightarrow Post(A')$; and (6) the role hierarchy of P′ is isomorphism to the role hierarchy of P.

A workflow process can include several sub-processes. The specialization relationship between two workflow processes can be further extended based on Definition 2 as follows.

**Definition 3.** If there exists a sub-process $p$ of P, and $p$–I → P′, then P′ is called a partial-specialization from P, denoted as P–P → P′. If there exists a sub-process $p'$ of P′, and P–I → $p'$, then P′ is called an extension-specialization from P, denoted as P–E → P′. If there exists a sub-process $p$ of P and a sub-process $p'$ of P′, such that $p$–I → $p'$, then P′ is called a revision-specialization from P, denoted as P′–R → P. If there does not exist a sub-process $p$ of P and a sub-process $p'$ of P′, such that P–I → P′, then P′ is called a non-specialization from P, denoted as P′–\ → P.

The sub-process mentioned in Definition 3 can further embed sub-processes. Definitions 2 and 3 define a multi-valued process specialization relationship between workflow processes, which can be used to realize a flexible workflow process reuse. Among these values of workflow process 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 partial-extension specialization is stronger than the non-specialization. As a representative, we show the example of the extension-specialization

Table 1
Rules for process specialization

| Rule number | Rules |
|---|---|
| 1 | $P_1–\gamma \to P_2$, $P_2–\gamma \to P_3 \Rightarrow P_1–\gamma \to P_3$ where $\gamma \in \{I,P,E\}$ |
| 2 | $P_1–I \to P_2 \Rightarrow P_1–P \to P_2$ |
| 3 | $P_1–I \to P_2 \Rightarrow P_1–E \to P_2$ |
| 4 | $P_1–I \to P_2 \Rightarrow P_1–R \to P_2$ |
| 5 | $P_1–E \to P_2 \Rightarrow P_1–R \to P_2$ |
| 6 | $P_1–P \to P_2 \Rightarrow P_1–R \to P_2$ |
| 7 | $P_1–I \to P_2$, $P_2–P \to P_3 \Rightarrow P_1–P \to P_3$ |
| 8 | $P_1–I \to P_2$, $P_2–E \to P_3 \Rightarrow P_1–E \to P_3$ |
| 9 | $P_1–I \to P_2$, $P_2–R \to P_3 \Rightarrow P_1–R \to P_3$ |
| 10 | $P_1–P \to P_2$, $P_2–R \to P_3 \Rightarrow P_1–R \to P_3$ |
| 11 | $P_1–E \to P_2$, $P_2–R \to P_3 \Rightarrow P_1–R \to P_3$ |
| 12 | $P_1–R \to P_2$, $P_2–\backslash \to P_3 \Rightarrow P_1–\backslash \to P_3$ |

in Fig. 2, where we have $A_i \to A'_i$ holds. If the workflow process P is available, we can reuse it when constructing process P′.

## 3. Activity-ontology and workflow-processes-ontology

Ontology usually contains a hierarchy of concepts on a domain and describes each concept's crucial properties through an attribute-value. It has been used to establish a certain common understanding between information-processing mechanisms in some domains. People have developed the conceptual-level ontology WordNet (www.cogsci.princeton.edu/~wn) and the assistant tools for the creation and management of ontology.

The activity-ontology repository defines a set of activity specialization graphs (ASG). In each ASG, nodes are activities, and arcs are activity specialization relationships. The elements of an activity can be specified by using the Extensible Markup Language (XML, see www.w3.org/TR/REC-xml), and the arcs can be specified by using the XLink. The basic management operations on the ASG include appending and deleting activity nodes. To append a new activity to an ASG needs to append the activity node and to establish the specialization relationship with its specialization predecessors and successors. To delete an activity node needs to delete the node and all the specialization arrows related to it. We have implemented these XML-based operations in our newly developed Knowledge Grid systems (see http://kg.ict.ac.cn).

The workflow-process-ontology repository defines a set of process specialization graphs (PSGs). Each PSG is a graph in which nodes are processes, and arcs are the multi-valued specialization relationships between the processes. A process in a PSG consists of the activities in the ASGs defined in the activity-ontology. The basic operations on PSG include node appending operation and node deleting operation. To append a new process to a PSG needs to append the node and to establish the multi-valued activity specialization relationship with its specialization predecessors and successors. To delete a process node needs to delete the node and all the specialization arrows related to it.

## 4. Theory for flexible process matching

### 4.1. Activity-distance and matching between activities

Let $A_Q$ and $A_C$ be two nodes in ASG ($A_Q$ can be a query specification). There are three possible relationships between $A_Q$ and $A_C$ on ASG: (1) $A_Q$ is more special than $A_C$; (2) $A_Q$ is the same as $A_C$; and (3) $A_Q$ is more general than $A_C$ (i.e. $A_C$ is more special than $A_Q$).

**Definition 4.** Activity-distance between two activities $A_Q$ and $A_C$, $d(A_Q, A_C)$, is the number of specialization arrows on the longest specialization path from $A_Q$ to $A_C$ on ASG and satisfies:

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

The activity-distance uses the longest specialization path (i.e. the worst case) to reflect a comparative activity specialization degree from one activity node to another on the same activity specialization path. A longer activity-distance between two activities provides more candidates similar to them. To append or to delete an activity node may change the length of the longest specialization path between two activities, so the activity-distance needs to be re-computed if the longest path has been changed. Based on the activity-distance, we can define the matching degree between activities.

In the following definition, we use a variable $s$ to control the search direction for the matching and a function $m$: $[0,1] \to [0,1]$ such that $m(0) = 0$, $m(1) = 1$, and for any $x$, $y \in (0,1)$, if $x \geq y$, then $m(x) \geq m(y)$ hold.

**Definition 5.** The matching degree between two activities $A_Q$ and $A_C$, denoted as $\mathrm{Match}_A(A_Q, A_C, s)$ is defined as: $\mathrm{Match}_A(A_Q, A_C, s) = m(1/(|d(A_Q, A_C)| + 1))$.

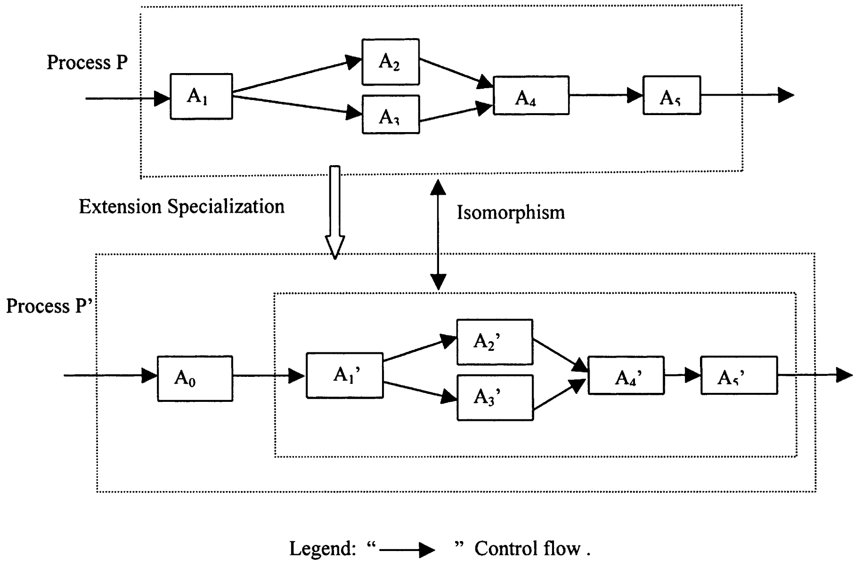


Fig. 2. An example of workflow process specialization.

### 4.2. Computing the similarity between two processes

Let $|P_Q|_A$ and $|P_C|_A$ be the number of activities included in $P_Q$ and $P_C$, respectively, and $|P_Q|_p$ and $|P_C|_p$ be the number of sub-processes included in $P_Q$ and $P_C$, respectively. *MF* is a one-to-one and onto match-making function that enables any activity $A_Q$ of $P_Q$ to match an activity of $P_C$ (i.e. $MF(A_Q)$) such that $d(A_Q, MF(A_Q))$ is the biggest of all the candidates. Variable $s = 0, 1$ and $-1$, respectively, denote the 'exact matching', '$P_Q$ is more general than $P_C$', and '$P_Q$ is more special than $P_C$'. Since $\text{Match}_A(A_Q, A_C, s) \in [0,1]$, we have $sd(P_Q,P_C,s) \in [0,1]$. If both $P_Q$ and $P_C$ have only one activity (i.e. $P_Q = \{A_Q\}$ and $P_C = \{A_C\}$), then $sd(P_Q, P_C, s) = \text{Match}_A(A_Q, A_C, s)$. $p_Q$ and $p_C$ are the sub-processes of $P_Q$ and $P_C$, respectively.

**Definition 6.** The similarity degree (*sd*) between two processes $P_Q$ and $P_C$ is defined with respect to the following two cases: Case1: $P_Q$–I → $P_C$; and Case 2: $P_Q$–I → $P_C$ and, $P_Q$–P → $P_C$, $P_Q$–E → $P_C$, or $P_Q$–R → $P_C$.

$$sd(P_Q, P_C, s) = \begin{cases} \sum \text{Match}_A(A_Q, A_C, s)/|P_Q|_A, \text{for all } A_Q \in P_Q, A_C = MF(A_Q) & \text{case 1;} \\ sd(p_Q, p_C, s) \times \varphi(|P_Q|_p, |P_C|_p, |P_Q|_A, |P_C|_A), & \text{case 2;} \end{cases}$$

The function $\varphi$ can be simply defined as: $(2/(|P_Q|_p + |p_C|_p) + (|p_Q|_A + |p_C|_A)/(|P_Q|_A + |P_C|_A))/2$. The similarity degree *sd* reflects the degree of two processes sharing activities or sub-processes in an isomorphism way.

**Definition 7.** Let $\beta \in [0,1]$ be an user expected value of the similarity degree (*sd*) between $P_Q$ and $P_C$, the process matching between process (or a query) $P_Q$ and process $P_C$ is defined as follows:

$$\text{Match}_p(P_Q, P_C, sd, s) = \begin{cases} \text{true}, & \text{if } sd \geq \beta; \\ \text{false}, & \text{otherwise.} \end{cases}$$

### 4.3. Rules for process specialization

Similar to the model inheritance rules proposed in Ref. [10], a set of rules for process specialization can be formed. Except for the revision-specialization, the other values of the process specialization have the transitivity characteristic represented as Rule 1 in Table 1. Rules 2–6 show the implication relationship between two different values of the process specialization. The transitivity and the implication characteristics enable different values of the process specialization to be connected for derivation. Rules 7–12 shown in Table 1 represent such a derivation.

Rule 1 can be proved according to Definition 2 and characteristic 1. 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. *Proof of Rule 7*: Since $P_1$–I → $P_2$ ⇒ $P_1$–P → $P_2$ (by Rule 2), we have: $P_1$–I → $P_2$, $P_2$–P → $P_3$ ⇒ $P_1$–P → $P_2$, $P_2$–P → $P_3$ ⇒ $P_1$–P → $P_3$ (by Rule 1).

The proposed rules in Table 1 can be used for: (1) establishing a rule reasoning mechanism for the heuristic process retrieval mechanism; (2) checking the consistency and the correctness of the specialization relationship and the similarity among the processes defined in the workflow-process-ontology repository; and (3) deriving a new matching from existing matchings.

### 4.4. Matching derivation

Matching derivation is to get a new matching directly from the existing matchings rather than to compute from scratch. We use '·' to denote the connection of two existing matchings, e.g. $Match_P(P_1, P_2, sd_3 \geq \beta_3, s) \cdot Match_C(P_2, P_3, sd_3 \geq \beta_3, s)$. 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 8.** Let $\gamma, \varphi \in \{I,P,E,\backslash\}$, $P_1-\gamma \rightarrow P_2$, and $P_2-\varphi \rightarrow P_3$. If there exist $\theta \in \{I,P,E,\backslash\}$ and a rule: $P_1-\gamma \rightarrow P_2, P_2-\varphi \rightarrow P_3 \Rightarrow P_1-\theta \rightarrow P_3$, then the connection of $Match_P(P_1, P_2, sd_1 \geq \beta_1, s)$ and $Match_P(P_2, P_3, sd_2 \geq \beta_2, s)$ is meaningful, otherwise is not meaningful.

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

$$Match_P(P_1, P_2, sd_1 \geq \beta_1, s) \cdot Match_P(P_2, P_3, sd_2 \geq \beta_2, s)$$

$$\Rightarrow Match_P(P_1, P_3, sd_3 \geq \beta_3 s)$$

where $\beta_3 = \nu(\beta_1, \beta_2)$ 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 the function $\nu$.

**Definition 9.** Let $\gamma, \varphi \in \{I,P,E,\backslash\}$, $P_1-\gamma \rightarrow P_2$, and $P_3-\varphi \rightarrow P_4$. If there exist: $\mu, \theta \in \{I,P,E,\backslash\}$, such that $P_2-\mu \rightarrow P_3$ and $P_1-\theta \rightarrow P_4$, then $Match_P(P_1, P_2, sd_1 \geq \beta_1, s) \cdot Match_P(P_3, P_4, sd_2 \geq \beta_2, s)$ is meaningful, otherwise is not meaningful.

If $Match_P(P_1, P_2, sd_1 \geq \beta_1, s) \cdot Match_P(P_3, P_4, sd_2 \geq \beta_2, s)$ is meaningful, a new matching can be formed by the following derivation:

$$Match_P(P_1, P_2, sd_1 \geq \beta_1, s) \cdot Match_P(P_3, P_4, sd_2 \geq \beta_2, s)$$

$$\Rightarrow Match_P(P_1, P_2, sd_1 \geq \beta_1, s) \cdot Match_P(P_2, P_3, sd_3 \geq \beta_3, s) \cdot Match_P(P_3, P_4, sd_2 \geq \beta_2, s)$$

$$\Rightarrow Match_P(P_1, P_4, sd_4 \geq \beta_4, s),$$

where $\beta_4 = \nu(\beta_1, \beta_2, \beta_3)$ such that $\beta_4 \leq \beta_i$ $(i = 1, 2, 3)$ and $\beta_4 \in [0,1]$ hold.

## 5. Inexact workflow process retrieval

### 5.1. Assistant tools for computing the similarity

We can use an activity-distance matrix ADM to record the initial values of the activity-distances between activities, denoted as $ADM = (Ad_{ij})_{n \times n}$, where $n$ is the number of the total activities in the activity-ontology repository, $Ad_{ij}$ represents the activity-distance between activity $A_i$ and activity $A_j$, denoted as $d(A_i,A_j)$, and satisfies: $Ad_{ii} = 0, Ad_{ij} = -Ad_{ji}$, and $Ad_{ij} \in (-\infty,\infty)$.

The process similarity degrees are recorded in a process specialization matrix (PSM), denoted as $PSM = (sd_{ij})_{e \times e}$, where $e$ denotes the total number of processes in the workflow-process-ontology repository, $sd_{ij}$ represents the similarity degree between $P_i$ and $P_j$, i.e. $sd(P_i,P_j,s)$, and satisfies: $sd_{ii} = 1$, $sd_{ij} = sd_{ji}$, and $sd_{ij} \in [0,1]$. ADM and PSM are used to avoid repeatedly computing a matching.

The development of the workflow-ontology-process repository consists of the following five steps:

1. describe every process and all the included activities;
2. establish the ASG and a relational table for ADM;
3. establish the PSG and a relational table for PSM;
4. design a maintenance mechanism for ASG, ADM, PSG, and PSM; and,
5. design the inexact retrieval mechanism.

The first step should be accomplished by the designers. Steps 2–3 can be accomplished by either the designers or the assistant tool. Users can adjust the contents in PSG and ASG with the help of a management mechanism. Steps 4–5 should be accomplished by the designers.

### 5.2. Inexact retrieval interface

The inexact process retrieval interface is an SQL-like language, which is similar to that we introduced in Refs. [10,13]. To avoid redundancy, we herein just present the different point. The workflow process specialization relationship is represented in the condition part of the query statement. The syntax of the condition portion is described as follows:

$$\langle Condition \rangle ::= P_1\text{–}I \rightarrow P_2 | P_1\text{–}P \rightarrow P_2 | P_1\text{–}E \rightarrow P_2 |$$

$$P_1\text{–}R \rightarrow P_2 | P_1\text{–}\backslash \rightarrow P_2 | \text{LIKE } P |$$

$$P_1 \langle P_2 | P_1 \rangle P_2 | P_1 = P_2 | PN = \langle String \rangle \sim | sd(P_1, ^*, s) \geq \beta |$$

$$\langle Condition \rangle AND \langle Condition \rangle | \langle Condition \rangle OR$$

$$\langle Condition \rangle | (\langle Condition \rangle)$$

where 'PN' denotes the process name. $PN = \langle String \rangle \sim$ means that '⟨String⟩' is the initial part of the name string of P. '$P_1 < P_2$' and '$P_1 > P_2$' mean that $P_1$ is more special and more general than $P_2$, respectively. The specialization hierarchy support a refinement retrieval strategy. Users can retrieve the required processes by using different conditions according to their familiarity degrees about the targets.

## 6. Discussion and comparison

The development of the activity-ontology repository and the workflow-process-ontology repository is domain-oriented and essential for realizing workflow process sharing. Human's participation is must during the development process although assistant tools can help to do some edit work and verification work.

Just as discussed in Ref. [10], a multi-valued process specialization hierarchy is helpful in raising the efficiency of the workflow process retrieval. On the other hand, a set of heuristic rules can be formed for search navigation, specialization relationship judgement, and the consistency checking between specialization relationships.

The proposed approach can be applied to the other fields to realize flexible reuse of various types of objects like models, software components, web services, web documents, knowledge, etc. The basis is the abstraction of these objects and the specialization relationships defined between them.

This work extends the approach [10] in the following three aspects. First, the function specialization is extended to the activity specialization, which concerns not only a function but also the restrictions and roles. Second, the definition of a workflow process is different from the definition of a model. Third, this paper proposes the theory for flexible process matching based on the similarity measuring between processes and the matching derivation. These extensions enable the proposed approach to be suitable for flexible workflow process reuse.

## 7. Conclusion

An inexact workflow process reuse approach has been presented. The main contribution of this work concerns: the workflow process abstraction and specification, the definition of workflow process specialization, the concept of activity-ontology and workflow-process-ontology, and the proposed theory for inexact process matching. The proposed approach can increase the flexibility of process matching by using the multi-valued process specialization relationship between the well-defined workflow processes. A matching candidate can be either an exactly required process or a relax one on the multi-valued specialization path with a quantified similarity degree of [0,1]. Based on the theory, we can implement the support tools for realizing process reuse when designing workflow processes.

Ongoing work concerns the following aspects: (1) investigating the characteristics of the reusable workflow processes; (2) the approach to integrate reusable workflow processes; (3) the approach to verify the correctness of the integrated workflow process; (4) apply the proposed approach to design web-service processes and the virtual organization business processes [12]; and (5) apply the proposed approach to realize knowledge-flow process management.

## Acknowledgements

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

## References

[1] D. Batory, S. O'Malley, The design and implementation of hierarchical software systems with reusable components, ACM Trans. Software Engng Methodol. 1 (4) (1992) 355–398.

[2] R.D. Banker, R.J. Kauffman, D. Zweig, Repository evaluation of software reuse, IEEE Trans. Software Engng 19 (4) (1993) 379–389.

[3] F. Leymann, D. Roller, Workflow-based applications, IBM Syst. J. 36 (1) (1997) 102–122.

[4] R. Mili, A. Mili, R.T. Mittermeir, Storing and retrieving software component: a refinement based system, IEEE Trans. Software Engng 23 (7) (1997) 445–460.

[5] E. Ostertag, J. Hendler, R.P. Diaz, C. Braun, Computing similarity in a reuse library system: an AI-based approach, ACM Trans. Software Engng Methodol. 1 (3) (1992) 205–228.

[6] J. Puustjarvi, H. Tirri, J. Veijalainen, Reusability and modularity in transactional workflows, Inf. Syst. 22 (2/3) (1997) 101–120.

[7] WfMC, The workflow Reference Model, http://www.wfmc.org.

[8] WIDE Consortium, WIDE Workflow Development Methodology, http://dis.sema.es/projects/WIDE/Documents/.

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

[10] H. Zhuge, Inheritance rules for flexible model retrieval, Decision Support Syst. 22 (4) (1998) 379–390.

[11] H. Zhuge, T.Y. Cheung, H.K. Pung, A timed workflow process model, J. Syst. Software 57 (3) (2001) 231–243.

[12] H. Zhuge, et al., An agent-workflow-federation approach for virtual organization development, Inf. Mgmt 39 (4) (2002) 325–336.

[13] H. Zhuge, A Knowledge Grid model and platform for global knowledge sharing, Expert Syst. Applic. 22 (4) (2002).