OWPA: An Ontology-Based Approach to Adaptable Workflow Participant Assignment

Jianmei GUO(a), Student Member, Yinglin WANG(b), and Jian CAO(c), Nonmembers

SUMMARY  Adaptable workflow participant assignment (WPA) is crucial to the efficiency and quality of workflow execution. This paper proposes an ontology-based approach to adaptable WPA (OWPA). OWPA introduces domain ontology to organize the enterprise data and uses a well-defined OWPA rule to express an authorization constraint. OWPA can represent more complex authorization constraints by flexibly using the enterprise data, the workflow data, the user-input data, and the built-in functions. By a high-usability interactive interface, OWPA allows users to define and modify the OWPA rules easily without any coding work. Moreover, OWPA is bound to the workflow modeling tool and the workflow monitor respectively to adapt to dynamic workflow modification in workflow definitions and workflow instances. OWPA has been applied in three enterprises in China.

key words: workflow, participant assignment, ontology, adaptability

1. Introduction

A workflow management system (WFMS) enables organizations to define, create, and manage business processes [25]. A business process is represented as a workflow in WFMSs. A workflow comprises a set of activities, which are units of work. An activity is performed by one or more workflow participants. The workflow participant is defined as a resource that performs the work represented by an activity [25]. It is normally applied to a human resource but it could conceptually include machine based resources such as an intelligent agent.

An efficient workflow should ensure that a given activity will be performed by the right workflow participant in due time [16]. However, the dynamic changes of practical workflow environment, such as department reorganization, absence of a workflow participant, always cause frequent workflow modification. Accordingly, a workflow demands the adaptability to workflow changes that can occur during workflow execution, i.e., the ability of dynamic modification [2], [20]. Dynamic workflow modification, depending on its durability, can be categorized as two types [16]: workflow definition modification concerns all related workflow instances and is usually associated with workflow optimization; workflow instance modification concerns some workflow instances and is usually associated with transient changes or exceptions.

Currently few WFMSs support dynamic workflow modification, including both workflow definitions and workflow instances. Most of the researches on dynamic workflow modification focus more on the control flow modification and less on workflow participant assignment (WPA) [16], [23]. Traditional WPA approaches mostly adopt role-based authorization and propose many specifications of different authorization constraints such as separation of duties [3], [5], [13]. They, however, are not adaptable to dynamic workflow modification, especially workflow instance modification.

Adaptable WPA is vital for most customer-centric organizations where many workflows are based on dynamic teams to serve customers with maximum satisfaction and minimal duration [1], [16], [18]. Some studies on adaptable WPA were presented [1], [7], [8], [16], but they are still insufficient. First, most of former approaches cannot sufficiently organize and use the enterprise data, the workflow data, and the user-input data in the WPA specification. Thus, some complex authorization constraints cannot be expressed yet. Second, former approaches mostly depend on a few professionals to define authorization constraints using complex specification languages. The specification often requires a great deal of programming work and is inconvenient to be modified.

This paper proposes an ontology-based approach to adaptable WPA (OWPA). OWPA introduces domain ontology to organize the enterprise data and uses a well-defined OWPA rule to represent an authorization constraint. OWPA represents more authorization constraints through flexibly using domain ontology, the workflow data, the user-input data, and the built-in functions. Moreover, OWPA flexibly defines and reconfigures an OWPA rule by a high-usability interactive interface. The interactive interface implements a context-aware mechanism that lists all candidate operands or operators automatically for the current editing OWPA rule. Thus, OWPA helps users to define OWPA rules easily without any coding. In addition, OWPA supports workflow definition modification and workflow instance modification in a unified manner.

The remainder of this paper is organized as follows. Section 2 presents the application background and describes a motivating example. Section 3 explains domain ontology and the organizational model as the foundation of OWPA. Section 4 proposes the definition of the OWPA rules. Section 5 details the implementation of OWPA. Section 6 gives...
a preliminary evaluation. Section 7 discusses our work and related work. Section 8 concludes the paper with future work.

2. Background and Motivation

We motivate this study because of the practical application of our ReKM (Reconfigurable Knowledge Management) system. ReKM supports collaborative knowledge management by integrating ontology and workflow techniques [21]. It has two main subsystems: an ontology management subsystem and a workflow management subsystem. The ontology management subsystem provides a standard ontological infrastructure (including classes, attributes, relations, and instances) for users to define their own domain ontology. All the enterprise data and knowledge are organized and managed in the domain ontology by an ontology modeling tool and a web portal. The workflow management subsystem implements collaborative knowledge processing such as knowledge capturing, reviewing, and codifying. It is designed on the basis of the WFMC workflow reference model [24] and thus comprises a set of standard workflow components such as a workflow modeling tool, a workflow engine, a workflow monitor, and an individual workbench.

Because of the inconstant enterprise requirements, ReKM develops a reconfigurable framework to support adaptable knowledge management in practice [22]. We have been studying the adaptability of ReKM to various changes of requirements that could be from data structures, interfaces, business processes, and integration environments [10], [11], [22], [23]. In this paper, we focus on adaptable WPA.

Three enterprises in China have applied ReKM for their design knowledge management. We hide their affiliation in order to honor confidentiality agreements. Consider, for example, a process of design knowledge review shown in Fig. 1, which is common to the three enterprises we investigated. A designer first summarizes his own design knowledge. Then the individual knowledge is reviewed by the right team leaders and domain experts. The approved design knowledge is recommended as the team knowledge and the organizational knowledge, and is further stored in enterprise knowledge base for later reuse in collaborative product design. The above review process is generally shared by all designers to deal with all types of design knowledge. Thus, different designers require different team leaders and different domain experts. Especially, the assignment of domain experts also depends on the type of the processing design knowledge. However, existing WPA approaches cannot represent the correlation between domain experts and the design knowledge being processed in workflows. Moreover, as a "scarce" resource, the predefined domain expert is often absent from his review activity during the execution of the review process. Thus, an efficient WPA approach must support the timely workflow instance modification to guarantee the efficiency of workflow execution. In addition, appropriate assignment of domain experts even affects the qualities of current knowledge review and later knowledge reuse. Therefore, OWPA is proposed.

3. Foundation of OWPA

3.1 Domain Ontology

Domain ontology is a formal and explicit specification of a shared conceptualization of a domain of interest [9]. ReKM organizes and manages the enterprise data by domain ontology. According to their specific requirements, enterprise users customize their domain ontology through the ontological infrastructure of ReKM. The ontological infrastructure conforms to the OI-model (ontology-instance-model) [17]. The OI-model adjusts the expressiveness of traditional logic-based ontology languages to sustain tractability [17]. An OI-model has an ontology structure associated with it, consisting of a set of definitions regulating how instances should be constructed. The ontology structure mainly consists of concepts (classes) and properties. Each property must have at least one domain concept, while its range may either be a literal (attributes), or a set of at least one concept (relations). Thus, the enterprise users can build their own domain ontology. Through domain ontology, OWPA can use the enterprise data flexibly to define the authorization constraints.

3.2 Organizational Model

Correct understanding and modeling organizational structures are the foundation of workflow participant definition [8]. Muehlen proposes an organizational reference metamodel to help users in specifying their requirements for a WFMS [18], as is shown in the right dashed box of Fig. 2. However, the “role” defined in Muehlen’s metamodel is relatively static and cannot satisfy adaptable WPA. We extend the metamodel by incorporating the “dynamic role”.

Fig. 1 A sample process of design knowledge review.

Fig. 2 An extended organizational model.
The dynamic role, shown in Fig. 2, is defined by the OWPA rules. Besides the organizational structure, the OWPA rules can describe more external factors such as customer expectations and regulatory aspects [18]. Thus, OWPA supports adaptable WPA through dynamic modification of organizational structure, especially the temporary or emergent modification.

Enterprise users can refer to the extended organizational model for modeling the organizational structure, and instantiate it when building domain ontology. The organizational model can be easily represented in domain ontology. For example, “Organizational unit” can be represented as a class, which has two subclasses “Department” and “Project”. It also has two relations “superclassOf” and “subclassOf” to represent its hierarchy.

4. Definition of OWPA Rules

4.1 Basic Elements

As is shown in Fig. 2, the OWPA rules comprise four basic elements: domain ontology, the workflow data, the user-input data and the built-in functions.

As mentioned in Sect. 3.1, the definition of domain ontology (DO) includes classes (Co), attributes (Ao), relations (Ro) and instances (Io). Generally, the top of the class hierarchy is indicated as the “Root”.

The workflow data (WfD) generally include three types: the control data represent the dynamic states of the WFMS and its process instances; the audit data represent the history of process instances execution; and the relevant data determine the state transitions of process instances. We categorize the WfD as two kinds: the workflow definition data and the workflow instance data. Some frequently-used workflow data are summarized in Table 1.

The user-input data (UiD) indicate the data input by users when specifying the OWPA rules. They support five basic data types: Boolean, Datetime, Integer, Double and String. They are often const values, e.g., integer “100”, string “Jack”, date “20080808”, or boolean “true”.

The built-in functions (BiF) are used to simplify the workflow data and the built-in functions. They are often const values, e.g., integer “100”, string “Jack”, date “20080808”, or boolean “true”.

The following table lists the set of attributes for a specific class; ‘*’ lists the set of attributes for a specific class or instance; ‘?’ finds the set of relations owned by a specific class or instance; ‘?’ indicates the range of a specific relation with some class or instance.

The relational operators: ‘==’, ‘<’, ‘<’ and ‘>’ and four new ontological relational operators. ‘o’ selects the set of instances for a specific class; ‘t’ lists the set of attributes for a specific class or instance; ‘?’ finds the set of relations owned by a specific class or instance; ‘?’ indicates the range of a specific relation with some class or instance.

4.2 Syntax

We define a language for the specification of OWPA rules (WRul). An OWPA rule comprises one or more OWPA formulas (wpa1) combined with the logic operators. An OWPA formula contains one or more OWPA expressions (wpa2) combined with the relational operators. The OWPA elements are composed of domain ontology, the workflow data, the user-input data, and the built-in functions. The syntax written in BNF is as follows:

Symbols used:

- ⟨UiD⟩, ⟨WfD⟩, ⟨BiF⟩, ⟨Co⟩, ⟨Io⟩, ⟨Ao⟩ and ⟨Ro⟩ have been explained in Sect. 4.1.
- The logic operators: ‘NOT’, ‘AND’ and ‘OR’.
- The algorithm operators: ‘+’, ‘-’, ‘*’, ‘/’ and four new ontological operators.
- ‘i’ selects the set of instances for a specific class; ‘t’ lists the set of attributes for a specific class or instance; ‘?’ finds the set of relations owned by a specific class or instance; ‘?’ indicates the range of a specific relation with some class or instance.
- The relational operators: ‘==’, ‘<’, ‘<’ and ‘>’ and a new ontological relational operator. ‘⊕’ is created to represent richer semantic relations. The r ∈ Ro is any relation defined in domain ontology. ‘⊕’ denotes that two classes or instances on both sides possess the relation r. For example, ‘Jack ∪ belongTo IT Dept.’ means the employee ‘Jack’ and the department ‘IT Dept.’ have the relation ‘belongTo’.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Examples of frequently-used workflow data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Name</td>
</tr>
<tr>
<td>Workflow definition data</td>
<td>ProcInit</td>
</tr>
<tr>
<td></td>
<td>FirstAct</td>
</tr>
<tr>
<td></td>
<td>LastAct</td>
</tr>
<tr>
<td></td>
<td>PrevAct</td>
</tr>
<tr>
<td></td>
<td>NextAct</td>
</tr>
<tr>
<td>Workflow instance data</td>
<td>CurrProc</td>
</tr>
<tr>
<td></td>
<td>CurrAct</td>
</tr>
<tr>
<td></td>
<td>ActPerf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Examples of frequently-used built-in functions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Name</td>
</tr>
<tr>
<td>Value-picking functions</td>
<td>ALL(S)</td>
</tr>
<tr>
<td></td>
<td>ONE(S)</td>
</tr>
<tr>
<td></td>
<td>MAX(S)</td>
</tr>
<tr>
<td></td>
<td>MIN(S)</td>
</tr>
<tr>
<td>Real-time analysis functions</td>
<td>AVAILABLE(P)</td>
</tr>
<tr>
<td></td>
<td>WORKLOAD(p)</td>
</tr>
</tbody>
</table>
4.3 Examples

Although the BNF language is simple and easy to understand, it still requires the users with a little programming background to edit the OWPA rules. The OWPA rules can represent most of the authorization constraints for practical WPA. Some examples are as follows. Here, we define the ‘GP’ (Goal Participants) to indicate the final participants accomplishing the goal of some activity. ‘GP’ is generally a set of the instances of class ‘Employee’.

Example 1: The ‘GP’ is an employee who has the same role with the performer of the previous activity but not the performer himself, which is a typical Separation-of-Duty authorization constraint scenario in practice.

\[ WRule\ 1: \\text{ONE} (GP \leftrightarrow CurrAct.\ PrevAct.\ ActPerf \\text{AND} GP \cdot \text{hasRole} \leftrightarrow CurrAct.\ PrevAct.\ ActPerf \cdot \text{hasRole}) \]

Example 2: The ‘GP’ are the employees who have the name ‘Jack’, or the employees who are available and the ‘Java’ expert with at least three-year experience, or the employees who belong to the same department with the performer of the first activity and have the workload not exceeding 3.

\[ WRule\ 2: \ (GP \leftrightarrow \text{Employee} \circ \text{“Jack”}) \\text{OR} \ (\text{AVAILABLE}(GP \cdot \text{hasSkills} \leftrightarrow \text{“Java” AND GP \cdot \text{hasServiceYears} \geq 3}) \ \text{OR} \ (GP \oplus \text{belongsTo} \otimes \text{Department} \leftrightarrow \text{CurrProc.FirstAct.\ ActPerf} \oplus \text{belongsTo} \otimes \text{Department} \ \text{AND} \ \text{WORKLOAD}(GP) \leq 3) \]

Example 3: Consider the motivating example described in Sect. 2: the ‘GP’ are the domain experts whose responsible knowledge domain covers the type of the design knowledge being processed in the previous activity. As is shown in Fig 1, we need to represent the correlation between domain experts and the processing knowledge, i.e., the cover relation of their knowledge domain (KDomain).

\[ WRule\ 3: \ GP \cdot \text{hasRole} \leftrightarrow \text{“Expert” AND ((GP \oplus \text{responsibleDomain} \otimes \text{KDomain}) \oplus \text{superclassOf}(\text{CurrAct.\ PrevAct.\ DesignKnowledge} \oplus \text{inDomain} \otimes \text{KDomain}))} \]

5. Implementation of OWPA

5.1 Implementing Framework

We implement an OWPA module in ReKM to define and execute the OWPA rules. As is shown in Fig. 3, the OWPA module has two components: the interactive interface allows users flexibly define and modify the OWPA rules; the interpretation interface interprets the OWPA rules and converts them to SQL. The components of ReKM, described in Sect. 2, provide the data source for the OWPA module.

We bind the OWPA module to the workflow modeling tool and the workflow monitor respectively for supporting workflow definition modification and workflow instance modification in a unified manner. First, in the workflow modeling tool, the OWPA module is used to define the activity performers. When the predefined performers need to be adjusted due to the possible process optimization, users can invoke the OWPA module to redefine the conditions of the right performers. Second, when an exception takes place and stops a running workflow instance, the workflow monitor will capture and analyze the exception. If the exception is a WPA exception (e.g., the predefined activity performers are unavailable or absent), the OWPA module will be invoked immediately to reassign the appropriate substitutes.

5.2 Interactive Interface

According to the OWPA syntax described in Sect. 4.2, we implement a high-usability interactive interface to help users define the OWPA rules easily without any coding. Through the interactive interface shown in Fig. 4, enterprise users can choose the right operands and operators to edit the OWPA rules. We explain the generation of the OWPA rules in Algorithm 1-3. Logically, an OWPA rule is formed by the OWPA formulas and the logical operators (see Algorithm 1). The OWPA formulas are built through selecting the right operand or operator alternately (see Algorithm 2).

To improve the usability of the interactive interface, we implement a context-aware mechanism that can list all available candidate operands or operators automatically for the next editing position of the current OWPA rule. A simplified version of the mechanism is described in Algorithm 3 where the workflow data and the built-in functions are not
For clearer description, Algorithm 3 uses the word ‘type’ to express the basic data types (‘Boolean’, ‘DateTime’, ‘Double’, ‘Integer’ and ‘String’) and the ontological types (‘Co’, ‘Io’, ‘Ao’ and ‘Ro’). In order to support the interactive selection of ontological operands, a set of ontological manipulations need to be provided. The minimal set is recommended in Table 3.

Table 4 exemplifies the generation process of the OWPA rule “GP • hasSkills == Java AND...”. In each step, users select an operand or operator (the underlined one shown in Table 4) to define the OWPA rule. When an operand or operator is selected and added to the current rule, the candidate elements for the next editing position will be changed dynamically according to the editing context and will be presented to users for the next selection. For example, when “GP” is selected, the next candidate operands ‘•’, ‘⊕’, ‘==’ and ‘<>' are presented according to Step 28 in Algorithm 3. When the formula “GP • hasSkills == Java” is completed, the logical operators ‘AND’ and ‘OR’ are also presented according to Step 7 in Algorithm 1. Note that the operand selection area and the operator selection area are

tabled dynamically according to the editing context and will be presented to users for the next selection. For example, when “GP” is selected, the next candidate operands ‘•’, ‘⊕’, ‘==’ and ‘<>' are presented according to Step 28 in Algorithm 3. When the formula “GP • hasSkills == Java” is completed, the logical operators ‘AND’ and ‘OR’ are also presented according to Step 7 in Algorithm 1. Note that the operand selection area and the operator selection area are included. For clearer description, Algorithm 3 uses the word ‘type’ to express the basic data types (‘Boolean’, ‘DateTime’, ‘Double’, ‘Integer’ and ‘String’) and the ontological types (‘Co’, ‘Io’, ‘Ao’ and ‘Ro’). In order to support the interactive selection of ontological operands, a set of ontological manipulations need to be provided. The minimal set is recommended in Table 3.

Table 4 exemplifies the generation process of the OWPA rule “GP • hasSkills == Java AND...”. In each step, users select an operand or operator (the underlined one shown in Table 4) to define the OWPA rule. When an operand or operator is selected and added to the current rule, the candidate elements for the next editing position will be

Table 3 The recommended ontological manipulations.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCo(Co)</td>
<td>Fetch all subclasses of Co</td>
</tr>
<tr>
<td>getIo(Co)</td>
<td>Fetch all instances of Co</td>
</tr>
<tr>
<td>getAoC(CoIo)</td>
<td>Fetch all attributes of Co or Io</td>
</tr>
<tr>
<td>getRoC(CoIo)</td>
<td>Fetch all relations of Co or Io</td>
</tr>
<tr>
<td>getAoT(Ao)</td>
<td>Fetch the basic data type of Ao</td>
</tr>
<tr>
<td>getRoC(IoRo)</td>
<td>Fetch all values of the range of Ro with Co or Io</td>
</tr>
</tbody>
</table>
6. Preliminary Evaluation

We obtained a preliminary evaluation of OWPA through organizing a symposium in some Bu Corporation where ReKM and OWPA have been applied. The symposium carried out a free discussion about the application of OWPA between the developers and the enterprise users. The enterprise users consisted of 15 members: one department manager, one domain expert, two project managers, five IT (information technology) supporters, three product designers, three quality controllers. Among them, five had the experience of workflow modeling; but the others had less or no programming experience. Their answers are collected and the results of the evaluation are summarized as follows.

Expressiveness. OWPA has more expressiveness of authorization constraints through sufficiently organizing and using the enterprise data, the workflow data, the user-input data, and the built-in functions. Especially, OWPA can represent the correlations between different data by defining the OWPA rules, e.g., Example 3 presented in Sect. 4.3. Thus some complex authorization constraints such as the motivating example described in Sect. 2, can be expressed.

Usability. OWPA provides a high-usability interactive interface to help users define the OWPA rules easily without any coding. It makes the specifying process of authorization constraints more convenient. Even after a short-term training, the users without any programming experience can participate in specifying the OWPA rules.

Limitation. Current OWPA still has some limitations that we plan to investigate further. First, OWPA alway asks users to define a complete OWPA rule. However, for some simple authorization constraints such as assigning a department role to the workflow participant, OWPA could be inconvenient because traditional role-based authorization only need to select a predefined role. Consequently, we have integrated OWPA and a simple role assignment tool in ReKM. Second, the current verification mechanism of the OWPA rules mainly depends on the syntax check and users’ experience. It cannot yet resolve the possible semantic conflicts for the editing OWPA rules.

In addition, although the high usability of OWPA makes users specify the OWPA rules more conveniently, it could generate some security risks and management problems to allow any user to participate directly in WPA. Thus, only some privileged users, involved in workflow modeling or human resource management, are allowed to use OWPA to specify the OWPA rules in practice.

7. Related Work and Discussion

Previous researches on dynamic workflow modification mostly focus on the control flow modification, and thus only a few approaches discuss adaptable WPA. Momotko and Subieta explained the dynamic changes of WPA and described the requirements of WPA for a modern WFMS [16]. They proposed a language named WPAL to handle WPA and its dynamic changes [16]. Deng et al. proposed an approach named ARDE to define the workflow participants from four facets, namely assignment, rule, duration and exception [8]: assignment uses an SQL-like language to define the qualification that activity participants must meet; rule expresses a regulation to choose users from all of qualified participants to undertake the task at runtime; duration is the interval during which the task starts and completes; exception represents the handling strategy when a task cannot be finished.

<table>
<thead>
<tr>
<th>Step</th>
<th>Candidate operators</th>
<th>Candidate operators</th>
<th>Current OWPA rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C, U, D, Cc, io, Ao, Ro, WFD, BIF</td>
<td>N/A</td>
<td>CP</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>$\cdot$, $\theta$, $\neq$, $&lt;$</td>
<td>CP $\cdot$</td>
</tr>
<tr>
<td>3</td>
<td>Ao</td>
<td>hasName, hasSkills, ..</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>$\geq$, $\leq$, $\geq$, $\leq$, $=$, $\neq$</td>
<td>CP $\cdot$ hasSkills $\geq$</td>
</tr>
<tr>
<td>5</td>
<td>C, U, D, Cc, io, Ao, Ro, WFD, BIF</td>
<td>String, Ecollate, integer, DateTime, Double</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>AND, CR</td>
<td>GP $\cdot$ hasSkills $\geq$Java AND</td>
</tr>
<tr>
<td>7</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 4 Demonstration of generating a sample OWPA rule.
in the planned duration. Cao et al. also proposed a task assignment policy language named TAPL to support dynamic process management [7]. However, the above approaches specify the workflow participants only using the workflow data and the organizational structure, which leads them to limited expressiveness. Moreover, they mostly obtain the adaptability to dynamic workflow modification by using the predefined functions. The modification of these hard-coding functions, however, always requires the professional developers and costs a great deal of programming work. Thus, the existing approaches to adaptable WPA cannot adapt to dynamic workflow modification efficiently.

Table 5 compares OWPA with the existing approaches to adaptable WPA. First, previous approaches cannot express all the enterprise data and the correlations between different data in the specification of WPA. OWPA introduces domain ontology to organize all the enterprise data. The well-defined OWPA rules can represent the correlations between different data. Second, previous approaches cost much programming work in specifying the authorization constraints and their changes. OWPA provides a high-usability interactive interface to help users specify the authorization constraints easily without any coding. It also improves the efficiency of dynamic workflow modification.

Kozaki et al. presented some consideration for dealing with roles using OWL, but they took a closer look at the semantic interoperability of ontologies [12]. Atluri et al. studied ontology-based workflow change management, but they focused on the decentralized workflow environment and the workflow migration consistency [4].

Osawa proposed a metalevel coordination strategy to implement an adaptive organization for reactive cooperative planning [19]. Borghoff et al. described an approach based on reflective workflow agents for adaptive workflows to manage the dynamic changes of business policies, but the reflective agents cannot persistently reuse the reflective abilities such as tracing rules [6]. WPA can be seen as a kind of reactive planning problem. Some researches on automated WPA were proposed recently [14], [15]. They analyze workflow event logs and mine participant assignment rules by machine learning techniques, but they suffer from limited application situations and lower accuracy.

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Comparison with related work on adaptable WPA.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison points</td>
<td>WPAL</td>
</tr>
<tr>
<td>Expressiveness</td>
<td>Expressing the workflow data</td>
</tr>
<tr>
<td></td>
<td>Expressing the enterprise data</td>
</tr>
<tr>
<td></td>
<td>Organizational structure</td>
</tr>
<tr>
<td></td>
<td>Expressing the correlations between different data</td>
</tr>
<tr>
<td>Usability</td>
<td>Using interactive interface to specify authorization constraints</td>
</tr>
<tr>
<td></td>
<td>Adapting to dynamic workflow modification without programming</td>
</tr>
</tbody>
</table>

8. Conclusion and Future Work

This paper proposes an ontology-based approach to adaptable WPA (OWPA). OWPA can represent more complex authorization constraints through flexibly using domain ontology, the workflow data, the user-input data, and the built-in functions. By the high-usability interactive interface, the enterprise users can specify the authorization constraints more easily without any coding. OWPA can adapt to dynamic workflow modification more efficiently.

In future, we will further improve OWPA to obtain better application, especially developing an automated mechanism for the consistency checking of the OWPA rules.

Acknowledgements

The authors thank the anonymous reviewers for their invaluable comments and the associate editor Kouji Kozaki for his excellent work. The authors also thank the reviewers and the attendees of BIR’08 for their helpful comments. This work was supported by the National Natural Science Foundation of China (NSFC No.60773088), the National High-tech R&D Program of China (863 Program No.2009AA04Z106), and the Key Program of Basic Research of Shanghai Municipal S&T Commission (No.08JC1411700).

References


Jianmei Guo is a Ph.D. student in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. He received the BS degree from Tianjin University and the MS degree from Shanghai Jiao Tong University. He had three-year working experience in the development and consultation of ERP systems. His research interests are software engineering and knowledge engineering, workflow management and knowledge flow management.

Yinglin Wang is a professor of the Department of Computer Science and Engineering at Shanghai Jiao Tong University. He received his Ph.D. degree from Nanjing University of Science and Technology in 1998. He was a visiting professor of Stanford University. His research interests include cooperative computing, ontology-based knowledge engineering and knowledge management. He has been in charge of multiple national research projects in China.

Jian Cao is the vice-director of the grid computing center and a professor of the Department of Computer Science and Engineering at Shanghai Jiao Tong University. He received his Ph.D. degree from Nanjing University of Science and Technology in 2000. His research interests include service computing, cooperative information system and software engineering. He has published more than 100 academic papers.