Multiagent and Grid Systems – An International Journal 5 (2009) 325–343
DOI 10.3233/MGS-2009-0132
IOS Press

Workflow management in the grid era: A goal-driven approach based on process patterns

Jinlei Jiang[a,b,*], Shaohua Zhang[a], Johann Schlichter[b] and Guangwen Yang[a]
[a]*Department of Computer Science and Technology, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, P. R. China*
[b]*Institut fuer Informatik, Technische Universitaet Muenchen, Boltzmannstr. 3, 85748 Garching, Germany*

Abstract. Workflow management in the grid era faces many challenges both related to workflow modeling and workflow execution. To deal with the challenges, the paper argues that the traditional process-driven workflow management paradigm should be replaced by a goal-driven one and proposes a corresponding implementation approach. The contributions of the paper are as follows: (1) a concept of process pattern is put forward that is an effective means not only for procedural knowledge representation and sharing but also for expertise exchange between domain experts and computer scientists; (2) based on process patterns, a goal-driven workflow framework is designed that incorporates semantic Web technologies and artificial intelligent (AI) planning techniques and that can (semi-)automatically generate a workflow on the fly according to the requirements of users and the running context.

Keywords: Process pattern, workflow generation, AI planning, ontology, grid computing

1. Introduction

Workflow technology, which has its origination in image processing in the early 1970s and which aims at the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to an overall business goal [43], has also evolved into the grid domain along with the rapid development of grid computing [21] due to its immense power in dealing with complex tasks and the integration of heterogeneous resources. Nowadays there are many experimental systems and products available, e.g. DAGMan [23], GridFlow [10], GridAnt [30], GWFE [45], Kepler [31], Taverna [35], and Triana [17], to name but just a few (refer to [46] for a taxonomy of existing systems). Still, there are more systems under development. Indeed, as pointed out in [40], large grid projects often opt to develop a new workflow tool rather than utilize or customize an existing solution.

Workflows in the grid domain are mainly utilized to support complex e-science applications such as climate modeling, high-energy physics, and disaster recovery simulation where a workflow process may comprise thousands of steps and each step might integrate diverse models and data sources that are

*Corresponding author. Tel.: +86 10 6279 6162; Fax: +86 10 6279 7141; E-mail: jjlei@tsinghua.edu.cn.

ISSN 1574-1702/09/$17.00

developed by different groups [18,25]. Usually this type of workflow is termed scientific workflow [13] to distinguish from business workflow. It is believed scientific workflows can accelerate the pace of scientific progress [18,25].

Although workflow systems have been demonstrated in a variety of scientific applications, there are still many requirements and challenges to be met before the promise of workflow technologies is fully realized [18,25]. In our opinion, though reproducibility [18,25] and security [14,22] are also important to scientific workflows, the core issue facing scientific workflow management for the time being is how to adapt the process/system to the changes and/or exceptions of business operations as well as the running environment.

It is a lasting effort in the business domain to handle changes/exceptions with workflow systems. As a result, much research work [2,3,8,11,20,28,38] has been reported. Though these existing approaches can meet the requirement to some extent in small-scale, centralized and homogeneous environments, they are generally not applicable to large-scale, distributed and heterogeneous environments like grid. The reason lies in that most existing workflow management systems (WfMSs), both in the business and the grid domain, adopt a process-driven paradigm, namely, a workflow process is firstly built by human designers usually and then submitted to an workflow engine for execution. Given the case of scientific workflow – both the experimental context of the user and the distributed infrastructure that the workflows operate over are in flux, such a paradigm, on the one hand, raises too high requirements on process designers – they should know not only the details of each business operation (e.g., pre-conditions, inputs, outputs, and post-conditions), but also the technical details of the process meta-model and the running environment (e.g., physical locations of resources, endpoints of services) to define a workflow process. On the other hand, it often leads to a quite complex and error-prone WfMS because requiring a WfMS to handle various changes/exceptions means increased complexity and this in turn means increased system errors. Therefore, the process-driven paradigm is not an ideal choice in the grid era.

To deal with the problem, this paper suggests a goal-driven paradigm be used instead and puts forward a new mechanism and the corresponding implementation framework. Here goal-driven means eliminating the boundary between workflow build-time and run-time. Instead of defining a complete workflow process in advance, users just specify a goal to achieve and then the WfMS will, with or without users' intervention, decompose it into some sub-tasks or steps according to the resources and knowledge possessed and fulfill the coordination and scheduling function necessary to achieve the specified goal. The decomposition procedure can be invoked several times along with the execution of the sub-tasks or steps identified, making it possible to tolerate and/or adapt to the changes and exceptions encountered meanwhile.

The rest of the paper is organized as follows. The following section gives an overview of the workflow framework proposed with a focus on system architecture and working procedure. Afterwards, details of process pattern, the core concept and the main contribution of this paper, and pattern-based workflow generation are explained in Section 3 and Section 4 respectively. Section 5 demonstrates the usage of the goal-driven workflow framework by a case study in logistics whereas Section 6 compares it with the related work. The paper ends in Section 7 with some conclusions where future work is also presented.

2. Towards goal-driven workflow management

This section reveals the fundamentals of the goal-driven workflow management and presents a brief yet full overview of the proposed implementation framework.

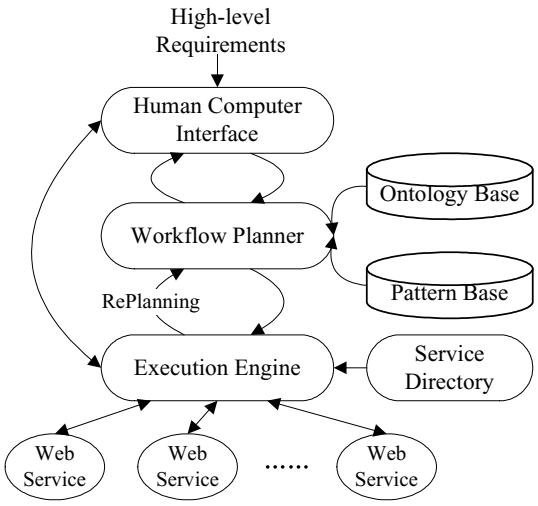

Fig. 1. The architecture of the proposed goal-driven workflow framework.

2.1. Theoretical foundation and design goal

The theory behind the goal-driven workflow management is situated action [39], which tells us the sequential organization of the task actions arises in response to the evolving immediate circumstances and exhibits a moment-by-moment, improvised character, rather than a predefined process as in process-driven WfMSs. However, taking situated actions as the underlying theory doesn't mean formal representation is not important. Indeed, this paper also favors the idea as stated by Bardram [6] – plans, as more or less formal representations, play a fundamental role in almost any organisation by giving order to work and thereby they effectively help getting the work done. It is in this sense that process pattern is put forward.

The goal-driven workflow management is possible because the progress in technology, especially the move to service-oriented architecture (SOA) and Web services [36] and the advance of semantic Web [7, 29] research, has established a solid technical foundation for it. Roughly speaking, SOA and Web services provide a flexible architecture and some reusable building blocks for constructing workflow systems and processes whereas the semantic Web provides the possibility for WfMSs to find desirable services automatically and precisely.

The goal-driven workflow framework design bears the following considerations in mind:

– It should balance the control flow between humans and automatic information systems to give enough respect for human proactiveness as well as to reduce system complexity. Full automation usually results in very complex and error-prone systems [27].
– It should support as many application domains as possible.

2.2. System architecture

Figure 1 illustrates the architecture of the proposed workflow framework. It can be divided into three layers: a) the upper layer is the Human Computer Interface through which end users can specify the goal to be achieved and intervene in the planning and execution phase to balance the control flow as aforementioned; b) the middle layer is the Workflow Planner. It is deployed to generate abstract

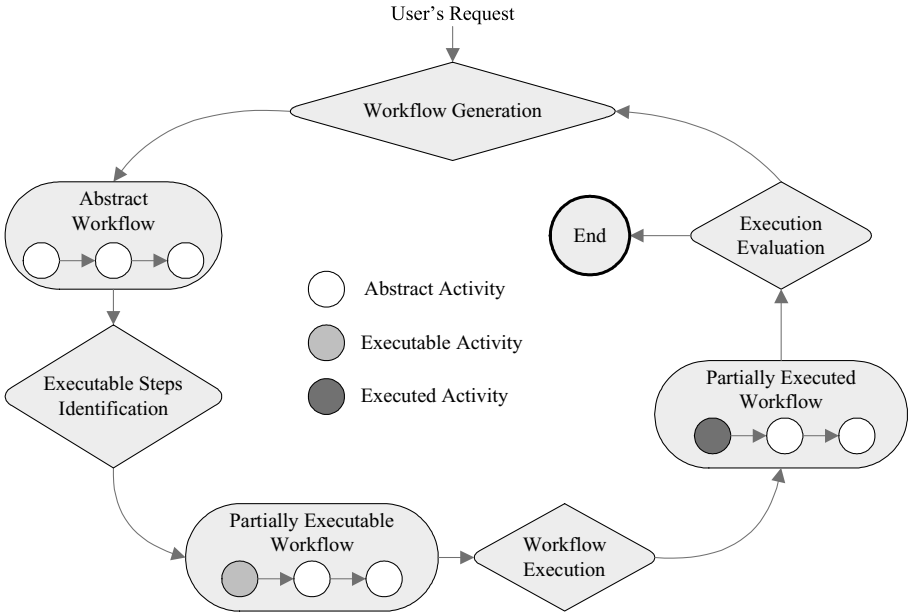

Fig. 2. A typical procedure for workflow construction and execution where some intermediate execution results of a certain loop are also displayed.

workflows and identify executable steps according to the goal specified, the current situation (context) and the knowledge stored in the ontology and pattern base; c) the bottom layer is the Execution Engine that maps steps of an abstract workflow to physical Web services and coordinates the execution of them.

In Fig. 1, the Ontology Base contains various domain ontologies that formally define the concepts used in a domain as well as the relationships between them. It functions as a basis for goal identification and decomposition. In this paper, OWL [34] is deployed to describe ontologies for it is a widely accepted standard. The Pattern Base stores various process patterns that form the basis for (semi-)automatic workflow generation. The Service Directory holds semantic-enriched specifications of various physical services. It functions like the information service in a grid and/or the service registry in an SOA environment.

2.3. Working procedure

The typical procedure for workflow construction and execution in the suggested goal-driven workflow framework is shown in Fig. 2 where each diamond represents a key algorithm. The procedure mainly consists of four steps as described below.

Step 1. Goal input

In this step, users specify a business goal (the User's Request in Fig. 2) via the Human Computer Interface.

Step 2. Workflow planning

In this step, the Workflow Planner first selects one or more process patterns from the Pattern Base to construct an abstract workflow (the Workflow Generation in Fig. 2) and then identifies some executable steps (the Executable Steps Identification in Fig. 2) according to the concepts in the Ontology Base and the current execution status. This step will be detailed further in Section 4.

Step 3. Workflow execution

In this step, the Execution Engine maps the identified steps to physical services and coordinates their execution (the Workflow Execution in Fig. 2).

Step 4. Execution evaluation

In this step, an evaluation algorithm (the Execution Evaluation in Fig. 2) is used to determine the successive actions: if the goal specified has been achieved, the procedure will end. Otherwise, it will restart the "planning-execution" loop from Step 2 until the final goal is achieved. This is just the meaning of the "RePlanning" arc in Fig. 1.

It is easy to see such a goal-driven approach enhances the flexibility of WfMS greatly because the steps to execute are determined on the fly that fully takes the current situation into account. In addition, other mechanisms introduced such as feedback (indicated by the control flow from the Workflow Planner/Execution Engine to the Human Computer Interface in Fig. 1) and user intervention (indicated by the control flow from the Human Computer Interface to the Workflow Planner/Execution Engine in Fig. 1) make the framework even more flexible.

3. Ontology and process pattern

The previous section has given an overview of the goal-driven workflow framework and pointed out that ontologies and process patterns form a foundation for (semi-)automatic workflow generation. This section further details them.

3.1. Requirements on knowledge representation

To deliver the proposed goal-driven workflow framework, knowledge is necessary. There are two types of knowledge related to workflow management, that is, environment knowledge and application-level knowledge. Environment knowledge depicts resources that constitute the running environment of WfMSs including their capabilities, policies of usage and the relationships between them. Application-level knowledge consists of business process expertise, user preferences, policy constraints, and other intelligence related to business operations. Environment knowledge is application-independent whereas application-level knowledge is application-specific.

Compared with environment knowledge, the value of application-level knowledge is underestimated. As workflows in the grid domain go beyond performing MPI/PVM jobs and storing huge data set to coordinate activities or services across organization boundaries, application-level knowledge should be paid more attention to because it plays a more significant role in coordinating workflow generation, execution and exception handling. The requirements on knowledge representation are as follows.

(1) It should be suitable for representing the procedural knowledge which describes the steps needed and their orders to fulfill a certain business task and which plays an important role to process design. In the past, such knowledge was usually stored in the brains of certain people and left out of the enterprise knowledge management systems.
(2) It should be effective and efficient. That's to say, the knowledge representation structure should meet the requirements of as many domains as possible. In addition, it must be apt to program and manipulate.
(3) It should be easy to understand so that user from the business field could smoothly accept it, use it and examine it.

Table 1
Key concepts of task ontology

Concept	Description
Description	It is a specification of the basic purpose to be achieved.
Deadline	It is the time constraints imposed on the task.
Group	It specifies the group (of person or software agent) in charge of the task.
Artifact	It is any entity (information, physical objects, etc.) that may be produced, consumed, or manipulated during the conduct of a task.
Attendee	It records the real participants of the task.

According to these criteria, some traditional knowledge representation techniques, like predicate logic, frames, semantic network, and rule-based method, are not suitable because rule-based method is not adequate for describing procedural knowledge, frames and semantic networks are hard to program, and predicate logic is too complex for users and business experts. Therefore, a more effective and easy to understand approach for knowledge representation is needed. This paper takes ontologies as the basis and proposes process patterns to fulfill the task of knowledge representation.

3.2. Ontology

Ontology is a formal model representing a set of concepts within a domain and the relationships between those concepts [41]. As the core concept of semantic Web [7,29], it can be used as a form of knowledge representation to enable knowledge sharing and reuse. This paper utilizes OWL [34] to describe ontologies. In this way, various domains of interest can be described uniformly, making the workflow framework independent of application domains.

Corresponding to environment knowledge and application-level knowledge, ontologies in this paper, though they share the same format, are also divided into two categories, namely environmental ontologies and domain ontologies.

Environmental ontologies define concepts common to different domains and/or applications. The design of environmental ontologies aims to facilitate the search for the appropriate process pattern and to provide the necessary interface and relationship information required for composing a single process of several process patterns. Currently the basic concepts identified include person, task, process, artifact, tool, policy and environment [42]. Among these concepts, task and process are two core concepts where task includes a description useful to users and to the workflow planner during execution and the process defines the concepts central to the management of business processes in general. The concepts of OWL-S [32] are borrowed to describe processes – either an atomic process which represents actions that can be performed by a single interaction at run-time, or a composite process which is composed of several atomic processes and/or composite processes and which represents work to be performed by multiple interactions. Each process, no matter it is atomic or composite, has properties like inputs, outputs, preconditions, results, etc. As for the key concepts of task ontology, they are listed in Table 1. Since ontology is not the emphasis, the paper will stop here. Please refer to [42] for the details.

Compared with environmental ontologies, domain ontologies cover a wide range of aspects. They play a very important role in automatic workflow generation. In this paper, domain ontologies are subdivided into two categories, namely service ontology and regulation ontology. Service ontology describes how a domain service is composed and can be viewed as the reification of process and task ontology in a specific domain. With service ontology, the Workflow Planner can learn the steps needed to fulfill a certain process/task. Regulation ontology defines the rules that guide domain operations. Example rules include “to get a Ph.D degree, one should have some papers published and pass the dissertation defense”,

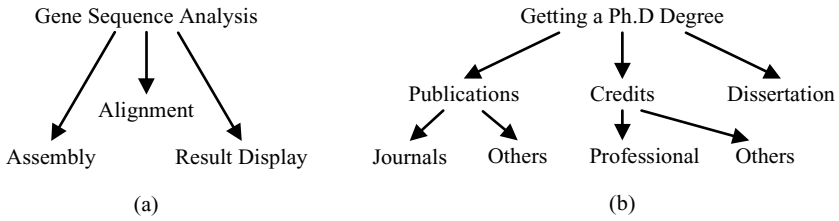

Fig. 3. Example domain ontologies: (a) service ontology for gene sequence analysis in bioinformatics; (b) regulation ontology for getting a Ph.D degree.

"one should be older than 18 to get a driving license", and so on. Figure 3 shows some example domain ontologies in a hierarchy format.

3.3. Process pattern

The term pattern originally comes from the architectural domain that "describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over" [4]. Favoring this concept, process pattern consists of three parts, namely, problem, context and solution where problem is a description of the task to be handled, and context specifies the conditions under which the solution will function, and solution provides a guideline to perform the task or gives an answer to the problem. Below is an example process pattern in XML format.

```
<Pattern>
 <Attribute type="id" value="4bec39c6-b727-4a2f-b83f"/>
 <Attribute type="name" value="Example Pattern"/>
 <Attribute type="domain" value="business.finance"/>
 <Attribute type="ontology" value=" http://..."/>
 <Attribute type="author" value="jjlei, zshua"/>
 <Attribute type="date" value="2007-04-13 09:09:10"/>

 <Problem>
   <Attribute type="function" value="Damage Claim"/>
   <Attribute type="keywords" value="Car"/>
   ...
 </Problem>

 <Context>
   <Item name="ClientType" RefValue="VIP"
         Operator="="/>
   <Item name="CreditLevel" RefValue="85"
         Operator=">"/>
 </Context>

 <Solution>
   <Transition type="Sequence">
```

```
 <Activity name="Act1" type="Standard">
   ...
 </Activity>
 <Activity name="Act2" type="Process">
  <Transition type="Sequence">
    <Activity name="Act21" type="Standard">
     ...
    </Activity>
    <Activity name="Act22" type="Standard">
     ...
    </Activity>
  </Transition>
  </Activity>
  <Activity name="Act3" type="Goal">
    <Attribute type="function" value="fConcept"/>
    <Attribute type="inputVariable" value="iNames"/>
    <Attribute type="outputVariable" value="oNames"/>
    <Inputs>?
      <Input name="paraName1" type="OntoConcept"/>+
      <InputMapping>Mapping XSLT</InputMapping>
    </Inputs>
    <Outputs>?
      <Output name="paraName2" type="OntoConcept"/>+
      <OutputMapping>Mapping XSLT</OutputMapping>
    </Outputs>
   </Activity>
  </Transition>
 </Solution>
</Pattern>
```

A process pattern starts with the keyword <Pattern> and six attributes where id is the internal identifier that uniquely identifies a process pattern, and name is user-friendly identifier for human computer interaction (to define or modify a process pattern), and domain specifies the application scope of the pattern, and ontology specifies the URI (Uniform Resource Identifier) of the referred ontology or ontology instance, and author records names of the authors involved in pattern definition, and date records the time that the process pattern was created.

The <Problem> section, consisting of a description of the function to achieve and one or more keywords about the problem, specifies a task to be handled and will be used by the Workflow Planner to map high-level user requirements to process patterns. To eliminate confusion, the function and the keywords of the task are defined with reference to domain ontologies specified by ontology attribute of this pattern.

The <Context> section specifies the situation under which the pattern will be applied. According to Dey et al. [19], context is "any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves." From this definition, it is obvious

that context involves nearly every aspect of an application and it is not feasible to define a generic context model that covers all the applications due to the diversity of applications. Therefore, an open implementation is adopted here that allows users to specify the conditions by defining one or more context items. Each context item, starting with the tag <Item>, has three attributes, i.e. name, RefValue and Operator where name indicates the name (e.g., ClientType and CreditLevel in the example) of context item according to which the runtime value will be retrieved, RefValue specifies a reference value for the context item which can be a concrete value as in the example or a set or an interval, and Operator is an operation symbol used by Workflow Planner to compare the current value of context item with RefValue. For the time being, operators supported include $>$ (greater than), $\geqslant$ (no less than), $<$ (less than), $\leqslant$ (no greater than), $=$ (equal), $\neq$ (inequal), in (within an interval or a set) and out (not within an interval or a set). A process pattern will function if and only if all the conditions specified by the context items are met.

The <Solution> section, starting with Transition which specifies the dependencies between activities indicated by Activity, provides an independent workflow fragment to solve the problem stated in <Problem> section. Currently dependencies supported include Sequence, Parallel Split, Synchronization, Exclusive Choice, and Simple Merge as stated in [1]. As for activities, three different types are distinguished: Standard, Process and Goal. The standard activity is an activity described using BPEL4WS specification [5] that can be directly recognized by any BPEL4WS-compliant execution engine (ActiveBPEL is exploited in this paper). The process activity, which is deployed to make the control flow more explicit and is composed of only standard activities here, is indeed a process with its own internal structure as in most WFMSs and as shown in the example pattern. The goal activity is deployed to describe those activities whose parameters can't be specified in advance. Details are as follows.

A goal activity, as shown in the example pattern, comprises the attribute function that points to some ontology entity (the fConcept as in the example) describing the functionality of this activity, the attribute inputVariable that points to the input variables (the iNames as in the example), the attribute outputVariable that points to the output variables (the oNames as in the example), and the semantic information related to inputs and outputs as defined by the Inputs and the Outputs respectively. Each input/output has two attributes with type pointing to a concept defined by an ontology (the OntoConcept as in the example) and name pointing to a concrete instance (the paraName1 and the paraName2 as in the example) of that concept. To describe the mapping between the XML schema based variables that involve no semantics and the semantic-rich OWL entities, InputMapping and OutputMapping are provided where InputMapping specifies how to map inputVariable to inputs and OutputMapping specifies how to map outputs to outputVariable. Both InputMapping and OutputMapping are described in the format of XSLT.

The three-part design of process patterns also facilitates the communication between domain experts who possess rich knowledge of the research domain but lack the knowledge of workflow language, the specifications of data movement, and computer scientists who, on the contrary, know well the detailed specifications of data movement and job execution steps but lack the knowledge of the research domain. When defining a process pattern, domain experts can specify details of the <Problem> and <Context> sections whereas computer scientists can finish details of the <Solution> part. Thus, the two views are linked together. Indeed, based on our previous work [26], a graphical definition tool is also supplied with which, on the one hand, designers can define various patterns simply by drag-and-drop and by specifying the properties needed as they do in most existing WfMSs and on the other hand, several designers can work collaboratively on the different parts of a process pattern as stated above. Such a way further eases the burden of pattern definition facing both domain experts and computer scientists.

3.4. Remarks on process patterns

Process pattern is a new way for knowledge representation. It is essentially different from workflow pattern [1] in that workflow pattern is a system-level technique to describe workflow processes that concerns only about the control structure of the workflow whereas process pattern is a high-level facility that takes workflow patterns as the basis to describe a process solution to a specific user task or goal under certain conditions. Problem and context are the essential part of a process pattern. Process patterns are also substantially different from predefined processes in a process library [44] in that these predefined processes provide no (explicit) information about the problem to solve and the conditions under which they will function whereas process pattern provides all.

In summary, process pattern has the following advantages for knowledge representation.

Firstly, process pattern is suitable for representing procedural knowledge. The three parts of a process pattern provide well-structured knowledge about how to solve a problem under certain conditions.

Secondly, process pattern is cost-effective. From users' perspective, a process pattern only explicitly associates a process with its goal and applied scenario. Therefore, it is convenient and easy to build and update the pattern base incrementally via the pattern definition tool supplied. From the system's perspective, the Workflow Planner mainly utilizes goal and context information to identify the appropriate solution. Pattern-based planning is simpler and more efficient than those methods that take all such details as operators, precondition, and effects into consideration from the beginning.

Thirdly, process pattern is easy to share. No confusion would arise since it is associated with public available and machine-readable ontology.

Fourthly, process pattern is fine-grained. It can describe knowledge of different abstract levels in various resolutions. Also, it provides adequate information about pattern application, making it possible for existing planners to generate workflow.

Fifthly and the last, process pattern is domain-independent. Though there is an attribute called "domain" in the pattern structure, it is only used to facilitate pattern selection. In other words, the structure of process patterns and the corresponding workflow generation algorithm are not adhered to a certain domain, making a good basis to broaden the application scope of the proposed workflow framework.

In the end, one point to highlight is that defining process patterns entails significant up-front developments costs. However, this initial development cost is more than offset by the potential ability for users to dynamically create their own workflows and the ability for WfMSs to adapt to run-time changes.

4. Pattern-based workflow generation

Based on (environmental and domain) ontologies and process patterns, a workflow can then be (semi-)automatically generated by means of an AI planning algorithm. Here "semi" means users may be involved (to provide information needed and/or to make a decision) during workflow generation. For the sake of simplicity, only automatic workflow generation will be explained here. In other words, the paper assumes there is at least one solution for each specified goal.

The procedure for automatic workflow generation is summarized as Algorithm 1. For the first time running, the algorithm needs to initialize partialFlow, a global tree structure deployed to store the intermediate planning results. The main program first searches the Pattern Base for the most suitable pattern that can achieve the goal g by invoking PatternMatching algorithm. If such a pattern exists, the corresponding solution is added to partialFlow and the algorithm proceeds to handle the goal activities

contained in the solution. Otherwise, a null solution is added to partialFlow and the algorithm deploys GoalDecomposition algorithm to decompose g into sub-goals (according to the corresponding ontology description) and does the planning procedure for each of them. In the end, different computed solutions are merged together (via SolutionComposition) to form a single execution plan. Note, due to insufficient information, the plan got may still contain abstract (i.e. goal) activities. Since the underlying execution engine can't recognize goal activities, the framework only propagates the standard executable activities (in BPEL4WS specification) to the engine for execution. It is up to the execution engine to guarantee the behaviour of the generated workflow. As pointed out in Section 2, the planning procedure may start again once all the identified activities have been executed.

```
Algorithm 1. WorkflowGeneration
Input: a business goal g
       a domain d
       a recursive level l
Output: An execution plan P
{
  if(l=0)
    partialFlow.Initialize();

  p ← PatternMatching(g, d);
  if(p≠null){//there is a solution for g in the pattern base
    partialFlow.AddNode(g, p.Solution);
    foreach(a∈P.Solution){
      if(a.type=Goal){
        p'← WorkflowGeneration(a.function, d, l+1);
      }
    }//end foreach
  }
  else{
    partialFlow.AddNode(g, null);
    G←GoalDecomposition(g);
    foreach(g'∈ G){
      p'← WorkflowGeneration(g', d, l+1);
    }//end foreach
  }

  P←SolutionComposition(partialFlow);

  return P;
}
```

PatternMatching, which is at the core of the WorkflowGeneration algorithm, is depicted in Algorithm 2. It first initializes P2, a link structure deployed to store the candidate process patterns. Then it goes to retrieve all the patterns related to the domain d using string matching (i.e. matching the domain attribute of a process pattern with d). Afterwards, the algorithm determines the patterns that can provide a solution for the goal g by checking whether the function attribute (in the <Problem> section) of the given process pattern is a sub-class of g (e.g. "Booking Train Ticket" is a sub-class of "Booking Ticket"). Such a comparison is called semantic matching. The patterns got in this step may still contain noises and therefore the current context is deployed to filter those patterns whose context items don't hold for the time being. In the end, the most suitable process pattern is selected (via PatternSelection) and returned. The knowledge specified by regulation ontologies will be utilized to calculate the suitability of a given process pattern against the running context.

5. A case study

Previous sections have detailed the key technologies behind the proposed goal-driven workflow framework. This section goes to demonstrate its usage by a case study in logistics.

```
Algorithm 2. PatternMatching
Input: a business goal g
       a domain d
Output: A pattern p that can achieve g
{
  P2 ← Φ;//a link structure

  P1 ← PatternQuery(d);//get all patterns related to the domain
  foreach(ptn∈P1){
    //the pattern provides a solution for g
    if(ptn.Problem.function⊆g)
      P2.Add(ptn);
  }

  cxt←getCurrentContext(g);//get the context related to g
  //Remove those patterns that can't function under current context
  P2 ← FilteringbyContext(P2, cxt);

  p ← PatternSelection(P2, g);
  return p;
}
```

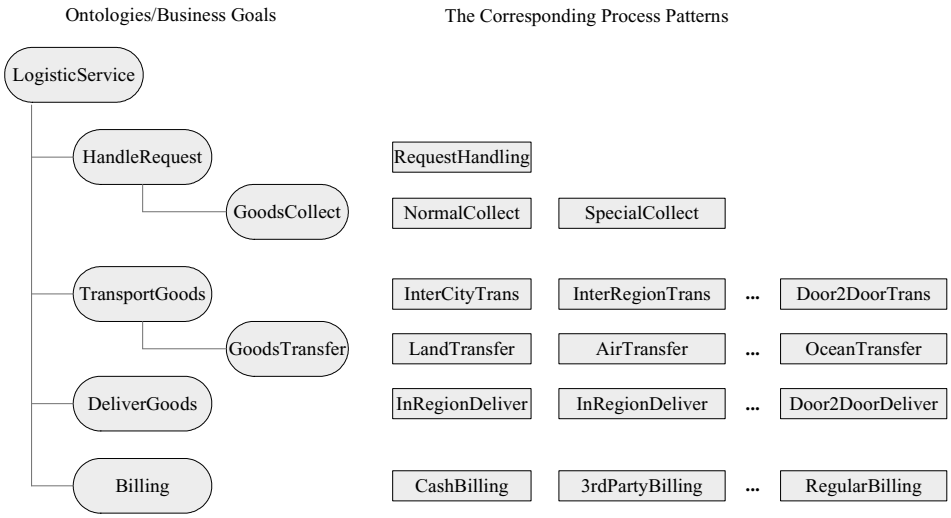

Fig. 4. Domain knowledge for logistical operations.

5.1. Background

Logistics is the discipline that manages the flow of raw material between the point of origin (source) and the point of consumption (destination). Though the purpose is simple, the processes handling the flow may vary greatly according to users' requirements and the attributes of the goods. For example, if the source and the destination are in the same area of a city, no transfer is needed. However, if they are in different cities, transfer is necessary. Moreover, according to the distance between the two cities, different transfer schemes may be deployed. In a word, logistics is dynamic! It involves many moving components that continually interact, constantly adding substance to an evolving process. Therefore, flexible process support is badly needed to reduce operation cost and to improve customer satisfaction. Such a requirement can be easily met by the proposed goal-driven workflow framework.

5.2. The goal-driven solution

Using the goal-driven workflow framework to support logistical applications, the only thing needed is to define domain knowledge using ontologies and process patterns. Since a domain can be described from different perspectives, the result ontologies and process patterns may vary. For most cases, there doesn't exist or it is hard to determine a best solution. Figure 4 shows one possible solution for logistics where only some top-level ontologies (defining business goals) and the corresponding process patterns are present for the sake of simplicity. In this solution, the supreme goal of logistics (i.e. LogisticService) is divided into four sub-goals, namely HandleRequest, TransportGoods, DeliverGoods and Billing. For each sub-goal, a set of process patterns are defined each of which provides a handling process that can fulfill the corresponding business goal under a certain scenario. As shown in Fig. 4, a sub-goal (e.g. HandleRequest or TransportGoods) may have its own subordinate goal(s).

5.3. An application scenario

From the end-user's perspective, there is no difference between using the goal-driven workflow framework and using the process-driven one. A user starts a logistical process by submitting a shipment request. Suppose the information got is as follows:

Source.UserName = `"Jinlei Jiang"`
Source.Street = `"FIT 3-113, Tsinghua University"`
Source.City = `"Beijing, China"`
Destination.UserName = `"Johann Schlichter"`
Destination.Street = `"Institut fuer Informatik, Boltzmannstr. 3"`
Destination.City = `"Garching, Germany"`
Goods.Category = `"Parcel"`
Goods.Name = `"Book"`
Goods.Weight = `500g`
Goods.Size = `50cm*30cm*20cm`
ExpectTime = `3 days`
PaymentType = `"Cash"`

Taking the information above as the running context, the procedure of workflow generation and the resulting process are shown in Fig. 5.

Details about this scenario are as follows. According to Algorithm 1, the system first searches the Pattern Base for the patterns that can fulfill the goal LogisticService. Since there exists no such patterns (please refer to Fig. 4), PatternSelection returns null. Therefore, after adding <g, null> to the tree (L0), the system decomposes g into four sub-goals (i.e. g1, g2, g3 and g4) and executes workflow generation algorithm for each of them (L1). Take g2 as an example, since Beijing and Garching are two different cities, the pattern InterCityTrans is selected. The <Solution> section of InterCityTrans contains a goal activity g21 and so the system continues executing the workflow generation algorithm on it (L2). This time AirTransfer is selected because the desired time delay (ExpectTime) is very tough for a distance as long as from Beijing to Garching. After all the (sub-)goals are unfolded, the function SolutionComposition is called to compose the desired process. The orders between the solutions are determined according to the corresponding service ontology.

From this scenario, it is easy to see the following two points. First, the goal-driven workflow framework can automatically generate task-specific processes. On the contrary, using the process-driven paradigm either means a lot of independent processes that have common steps or means a very complex process that

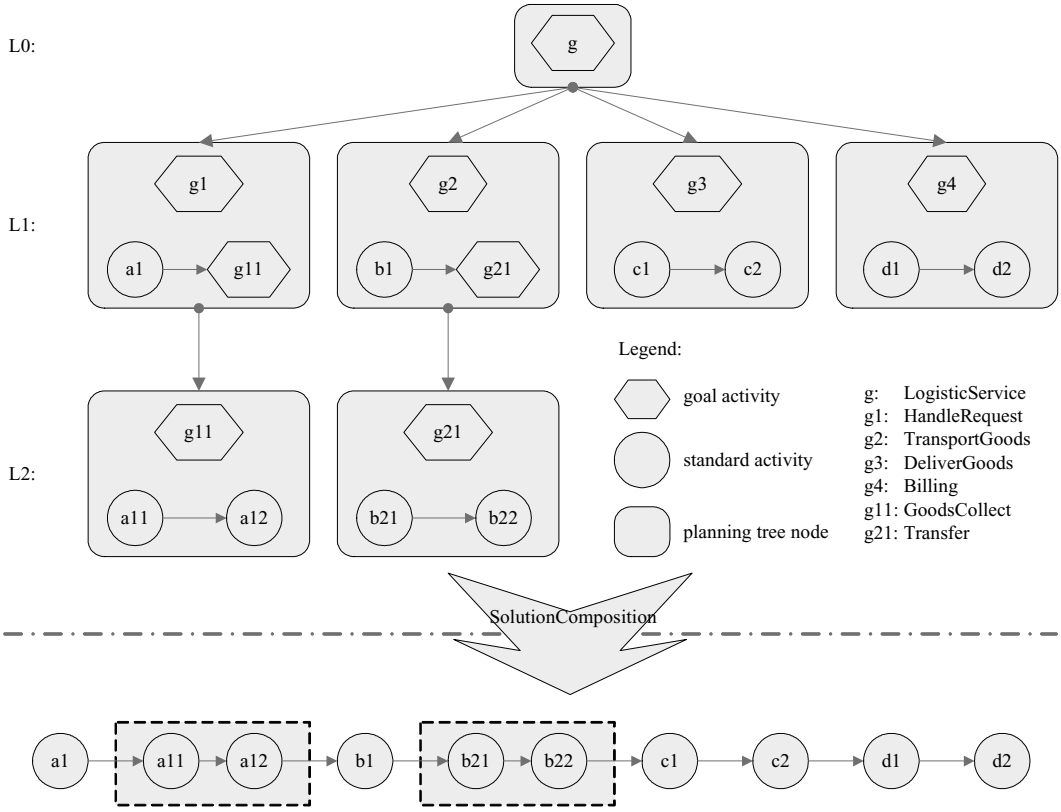

Fig. 5. The planning procedure and the result process for sending a parcel from Beijing, China to Garching, Germany.

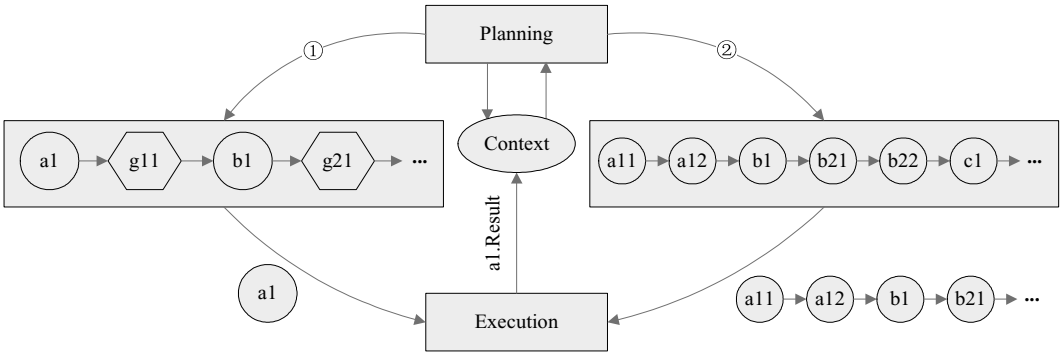

Fig. 6. The interleaving of process planning and process execution.

describes various conditions. Both ways are impractical. Second, a process can be easily reengineered by adding new or altering existing ontologies and/or process patterns. No modification to the underlying business systems is needed. This point is especially significant to improve the responsiveness of an enterprise.

Indeed, the system can get even more flexible by interleaving the planning procedure and process execution. Also take this scenario as an example and suppose a1 is an activity to analyze the information provided by the user, which means the solutions for g11 and g21 can't be determined at L1 planning.

Therefore the algorithm WorkflowGeneration returns a process with abstract activities. For this process, a1 is executable. After a1 is executed, the context is updated and the planning process restarts. As a result, g11 and g21 are unfolded and a new process is computed and submitted for execution. Figure 6 illustrates the whole process.

6. Related work

PROTEUS [9] is a grid-based problem solving environment where workflow technology is deployed for the design, schedule and control of bioinformatics applications. Though ontologies are deployed to model the semantics of application goals and requirements, it is not a goal-driven workflow management system because there is still an explicit boundary between workflow build-time and run-time. In PROTEUS, the knowledge depicted by ontologies is only used as a dictionary by the workflow design tool to assist users (searching available services, software widgets, data, etc.) in defining workflow processes rather than as a basis for automatic workflow generation and enactment.

Automatic workflow generation is also discussed in [15] as an alternative design paradigm to traditional manual one that relies heavily on humans to specify business processes statically. The authors take domain knowledge (represented as service ontology, regulatory ontology and a user profile) as the basis and adopt a rule rewriting approach to dynamically generate a workflow process. Compared with the approach presented here, the work in [15] is a fine-grained approach – users should specify all the conditions under which a task will function. As the application domain expands, the rule base will become quite large, resulting in low efficiency of workflow composition. In addition, the work in [15] only considers workflow generation whereas dynamic modifications of workflow definitions and adaptation to run-time changes/exceptions are left unresolved. As a result, its application scope is limited.

Pegasus [24] is a typical workflow system that presents some goal-driven features. In Pegasus, the only thing users need to do is specifying an application-level description of the desired data product. Taking the desired data products as the goals and the application components as the operators, Pegasus deploys an AI planning system to search for a valid, partially ordered set of operators that will transform the current state into one that satisfies the goal. The planning result corresponds to an executable workflow that can be transformed into a directed acyclic graph for execution over DAGMan [23]. The main problem with Pegasus is the lack of explicit knowledge representation – the planner and the knowledge used for planning are tightly bound to a specific application domain. Consequently, it can't be used to address more aspects of the grid environment's workflow management problem. To do so, as pointed out by the authors, a more distributed and knowledge-rich approach is required. The work reported here is such an approach where explicit knowledge representation (ontologies and process patterns) is deployed that makes the workflow planner domain-independent and extends the application scope of the proposed workflow framework.

The task based process management (TBPM) project [16] presents a similar approach. TBPM utilizes ontology-based models to describe not only the process itself but also the knowledge about application domain and organizational context where the process is enacted. Though it can facilitate workflow generation and enactment, the TBPM approach imposes a special requirement that domain ontologies must specialize the general concepts of the process ontology, making it difficult, if not impossible, to reuse ontologies that are defined without knowing the process ontology. This paper puts domain ontologies in the first place – not only the user's requests (goals) but also the process patterns are specified with reference to domain ontologies – making it possible to protect existing investigations, which is one of the keys to the success of a certain technology. In addition, plans in TBPM are only distinguished by

their types. As the application domain expands, it will become difficult to specify the types of tasks and to identify a suitable plan for a certain situation. In this paper, process patterns are distinguished by both the application domain and the task type (indicated by "description" and "keywords" in the <Problem> section), making it efficient and accurate to find a solution for a given situation even if multiple application domains are involved.

Plægine [33] is a software system that aims to support automated service compositions as well as their flexible enactment. To achieve the purpose, Plægine incorporates AI planning techniques into Web services composition. Drawbacks with Plægine are the following: 1) states are modelled as logical expressions, which means a burden to users because they have to learn to define goals in the planning language; 2) Planning and enactment are separated, which implies inability to adapt to the runtime changes especially for long-lived composite services. In this paper ontology is deployed to describe business goals and planning and enactment are interleaving. Therefore, the drawbacks above no longer exist.

AI planning techniques are also deployed in [12,37] to facilitate workflow management. The work in [12] is specially devised for portal-based workflow where the number of operators is limited and centralized system architecture is adopted. So, it can't be applied to grid domain where the number of operators is huge and peer-to-peer architecture is often used. The work in [37] tries to integrate IPSS, a planner that integrates planning and scheduling, with SHAMASH, a workflow modeling tool that allows non-experts entering knowledge on processes. However, it is not an easy task to translate the rich semantic representation of SHAMASH into IPSS planning language. Given the complexity of grid and various modeling tools available, the work in [37] can't be applied to grid domain either.

7. Conclusions and future work

This paper advocates a goal-driven workflow management paradigm in the grid era and proposes an implementation framework. Two steps adopted are as follows. Firstly, process pattern is proposed as an effective way for (procedural) knowledge representation as well as a means for domain experts and computer scientists to exchange expertise. Taking domain ontologies as the basis, process patterns can be shared and reused easily without causing any confusion. Secondly, a planning algorithm is designed that takes process patterns as the building blocks and that can compose (in partial or full) a workflow on the fly according to the goal specified by users and the running context. On account of the fact that full automation view of system design always leads the system hopelessly complex and error-prone [27], human intervention is introduced that not only gives enough respect for human proactiveness but also reduces system complexity.

As demonstrated by the case study in logistics, such a goal-driven workflow framework has at least the following benefits.

- From a system's perspective, it greatly enhances the flexibility and adaptability of the workflow system. Since there is no more explicit boundary between workflow build-time and run-time and the executable steps are determined on the fly according to the domain knowledge (expressed by domain ontologies and process patterns) and the running context, WfMSs can easily adapt to the runtime changes and exceptions. In addition, the support of user intervention in planning and execution balances the control flow between humans and automatic information systems, making the framework even more flexible.
- From end-users' perspective, it enormously eases the burden of process definition because the only thing needed to fulfill a task is to specify the goal (what needs to be achieved) rather than a plan (how the goal is accomplished) and thus process designers need to know neither the details of each

business operation nor the details of the process meta-model and the running environment. Given the fact that workflow users in grid domain are usually domain scientists who know little about workflow specification and job execution, this point is especially significant.

Though the goal-driven workflow framework presents quite some outstanding features, the work reported here is only the first step towards fully realizing the promise of workflow technologies. The future work includes: 1) investigating how the quality of domain knowledge influence the outcome of the Workflow Planner, and 2) investigating machine learning algorithms to capture the experience gained during process planning and execution so as to enrich domain knowledge while reducing the up-front development costs of domain ontologies and/or process patterns.

Acknowledgements

The authors thank the anonymous referees for their valuable comments on an earlier version to make this a better article. The work reported in this paper is co-sponsored by Alexander von Humboldt Fellowship, Natural Science Foundation of China under grant No. 60573110, 90612016, and 60673152, National High Technology Development Program of China under grant No. 2006AA01A108, 2006AA01A111, 2006AA01A101, and 2006AA01A117, and National Key Basic Research Project of China under grant No. 2003CB317007.

References

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros, Workflow Patterns, *Distributed and Parallel Databases* **1** (2003), 5–51.
[2] W.M.P. van der Aalst and S. Jablonski, Dealing with Workflow Change: Identification of issues and solutions, *International Journal of Computer Systems, Science, and Engineering* **5** (2000), 267–276.
[3] M. Adams, A.H.M. ter Hofstede, D. Edmond and W.M.P. van der Aalst, Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows, in: *Proc the 14th International Conference on Cooperative Information Systems, Lecture Notes in Computer Science* **4275** (2006), 291–308.
[4] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King and S. Angel, A Pattern Language, Oxford University Press, New York, 1977.
[5] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic and S. Weerawarana, Business Process Execution Language for Web Services, May 2003.
[6] J. Bardram, Plans as Situated Action: An Activity Theory Approach to Workflow Systems, in: *Proc the 5th European Conference on Computer-supported Cooperative Work*, 1997, pp. 17–32.
[7] T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web, *Scientific American* **5** (2001), 35–43.
[8] A. Borgida and T. Murata, Tolerating exceptions in workflows: a unified framework for data and processes, In: Proc International Joint Conference on Work Activities, *Coordination and Collaboration* (1999), 59–68.
[9] M. Cannataro, C. Comito, A. Guzzo and P. Veltri, Integrating Ontology and Workflow in PROTEUS, a Grid-based Problem Solving Environment for Bioinformatics, in: *Proc International Conference on Information Technology: Coding and Computing*, IEEE Computer Society Press, Vol. 2, 2004, 90–94.
[10] J. Cao, S.A. Jarvis, S. Saini and G.R. Nudd, GridFlow: Workflow Management for Grid Computing, in: *Proc the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society Press, 2003.
[11] F. Casati, S. Ceri, B. Pernici and G. Pozzi, Workflow Evolution, in: *Proc the 15th International Conference on Conceptual Modeling*, Lecture Notes in Computer Science 1157, 1996, pp. 438–455.
[12] M. Cheatham and M.T. Cox, AI planning in portal-based workflow management systems, in: *Proc International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005, pp. 47–52.
[13] J.J. Chen and W.M.P. van der Aalst, On Scientific Workflow, TCSC Newsletter 1, 2007, http://www.ieeetcsc.org/newsletters/2007-01/scientificworkflow.html.
[14] H. Chivers and J. McDermid, Refactoring service-based systems: how to avoid trusting a workflow service, *Concurrency and Computation: Practice and Experience* **10** (2006), 1255–1275.

[15] S.A. Chun, V. Atluri and N.R. Adam, Domain Knowledge-based Automatic Workflow Generation, in: *Proc the 13th International Conference on Database and Expert Systems Applications*, Lecture Notes in Computer Science 2453, 2002.
[16] P.W.H. Chung, L. Cheung, J. Stader, P. Jarvis, J. Moore and A. Macintosh, Knowledge-based process management – an approach to handling the adaptive workflow, *Knowledge-Based Systems* **3** (2003), 149–160.
[17] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor and I. Wang, Programming Scientific and Distributed Workflow with Triana Services, *Concurrency and Computation: Practice and Experience* **10** (2006), 1021–1037.
[18] E. Deelman and Y. Gil, Final Report for Workshop on Challenges of Scientific Workflows, 2006, http://vtcpc.isi.edu/wiki/images/3/3a/NSFWorkflowFinal.pdf.
[19] A.K. Dey, G.D. Abowd and D. Salber, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *Human-Computer Interaction* **2** (2001), 97–166.
[20] S. Ellis, K. Keddara and G. Rozenberg, Dynamic Changes within Workflow Systems, in: *Proc ACM Conference on Organizational Computing Systems*, 1995, pp. 10–21.
[21] I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International Journal of High Performance Computing Applications* **3** (2001), 200–222.
[22] G.C. Fox and D. Gannon, Special Issue: Workflow in Grid Systems, *Concurrency and Computation: Practice and Experience* **10** (2006), 1009–1019.
[23] J. Frey, T. Tannenbaum, M. Livny, I. Foster and S. Tuecke, Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Cluster Computing* **3** (2002), 237–246.
[24] Y. Gil, E. Deelman, J. Blythe, C. Kesselman and H. Tangmunarunkit, Artificial Intelligence and Grids: Workflow Planning and Beyond, *Intelligent System* **1** (2004), 26–33.
[25] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau and J. Myers, Examining the Challenges of Scientific Workflows, *Computer* **12** (2007), 24–32.
[26] J.L. Jiang and M.L. Shi, Agent Aided Workflow Modeling, in: *Proc Advanced Workshop on Content Computing*, Lecture Notes in Computer Science 3309, 2004, pp. 39–45.
[27] J.L. Jiang, S.H. Zhang, Y.S. Li and M.L. Shi, CoFrame: A Framework for CSCW Applications Based on Grid and Web Services, in: *Proc IEEE International Conference on Web Services*, 2005, pp. 570–577.
[28] P.J. Kammer, G.A. Bolcer, R.N. Taylor, A.S. Hitomi and M. Bergman, Techniques for Supporting Dynamic and Adaptive Workflow, *Computer Supported Cooperative Work* **3&4** (2000), 269–292.
[29] O. Lassila and J. Hendler, Embracing Web 3.0, *IEEE Internet Computing* **3** (2007), 90–93.
[30] G.V. Laszewski, K. Amin, M. Hategan, N.J. Zaluzec, S. Hampton and A. Rossi, GridAnt: A Client-Controllable Grid Workflow System, in: *Proc the 37th Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, 2004.
[31] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao and Y. Zhao, Scientific Workflow Management and the Kepler System, *Concurrency and Computation: Practice & Experience* **10** (2006), 1039–1065.
[32] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan and K. Sycara, OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S, 2004
[33] H. Meyer, H. Overdick and M. Weske, Plængine: A System for Automated Service Composition and Process Enactment, In: Proc Workshop on WWW Service Composition with Semantic Web Services, 2005, 3-12
[34] D.L. McGuinness and F. van Harmelen, eds, OWL web ontology language overview, http://www.w3.org/TR/owl-features, 2004.
[35] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat and P. Li, Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows, *Bioinformatics* **17** (2004), 3045–3054.
[36] M.P. Papazoglou and W. van den Heuvel, Service oriented architectures: approaches, technologies and research issues, *The VLDB Journal* **3** (2007), 389–415.
[37] M.D. R-Moreno, D. Borrajo, A. Cesta and A. Oddi, Integrating planning and scheduling in workflow domains, *Expert Systems with Applications* **2** (2007), 389–406.
[38] S.W. Sadiq, W. Sadiq and M.E. Orlowska, Pockets of Flexibility in Workflow Specification, in: *Proc the 20th International Conference on Conceptual Modeling, Lecture Notes in Computer Science* **2224** (2001), 513–526.
[39] L. Suchman, Plans and situated actions: The problem of human machine communication, Cambridge: Cambridge University Press, 1987.
[40] I. Taylor, M. Shields, I. Wang and A. Harrison, Visual Grid Workflow in Triana, *Journal of Grid Computing* **3** (2006), 153–169.
[41] M. Uschold and M. Gruninger, Ontologies: principles, methods and applications, *Knowledge Engineering Review* **2** (1996), 93–136.

[42] G.L. Wang, J.L. Jiang and M.L. Shi, Modeling Contexts in Collaborative Environment: A New Approach, in: Computer Supported Cooperative Work in Design III, *Lecture Notes in Computer Science 4402*, 2007, pp. 23–32.
[43] Workflow Management Coalition, The workflow reference model, 1995.
[44] G.X. Yang, Process library, *Data Knowledge and Engineering* **1** (2004), 35–62.
[45] J. Yu and R. Buyya, A Novel Architecture for Realizing Grid Workflow using Tuple Spaces, in: *Proc the 5th IEEE/ACM International Workshop on Grid Computing*, IEEE Computer Society Press, 2004, pp. 119–128.
[46] J. Yu and R. Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, *Journal of Grid Computing* **3** (2006), 171–200.

Authors' Bios

Jinlei Jiang is an assistant professor with Department of Computer Science and Technology, Tsinghua University, Beijing, China. From May 2007 to April 2008, he visited Institut fuer Informatik, Technische Universitaet Muenchen, Germany with the Group of Applied Informatics – Cooperative Systems under the sponsorship of Alexander von Humboldt Foundation. Jiang received a PhD degree in computer science and technology from Tsinghua University in January 2004 with an honor of excellent dissertation. His research interests mainly focus on CSCW, workflow management, grid computing and Web services. His research work has appeared in Expert Systems with Applications, Journal of Computer Science and Technology, Lecture Notes in Computer Science, IEEE International Conference on Web Services, ACM Symposium on Applied Computing and so on.

Shaohua Zhang is a full-time PhD candidate with Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests are grid computing, CSCW and intelligent systems. Contact him at zshua@csnet4.cs.tsinghua.edu.cn.

Johann Schlichter is a professor with Institut fuer Informatik, Technische Universitaet Muenchen, Germany, where he chairs the Group of Applied Informatics – Cooperative Systems (http://www11.in.tum.de/). His research interests include groupware, knowledge management, Web Applications/Web Services and Multimedia in Teaching. Contact him at schlichter@in.tum.de.

Guangwen Yang is a professor with Department of Computer Science and Technology, Tsinghua University, Beijing, China. Currently, he is on the expert committee of high performance computer of National High Technology Development Program of China and directs the Institute of High Performance Computing, Tsinghua University. Prof. Yang earned a PhD degree from Harbin Institute of Technology. His research interests are mainly on parallel computing, grid computing, distributed systems and machine learning. Contact him at ygw@tsinghua.edu.cn.