An agent-based Web service workflow model for inter-enterprise collaboration

Shuying Wang, Weiming Shen *, Qi Hao


Abstract

The service-orientated computing paradigm is transforming traditional workflow management from a close and centralized control system into a worldwide dynamic business process. A complete workflow serving inter-enterprise collaboration should include both internal processes and ad hoc external processes. This paper presents an agent-based workflow model to address this challenge. In the proposed model, agent-based technology provides the workflow coordination at both inter- and intra-enterprise levels while Web service-based technology provides infrastructures for messaging, service description and workflow enactment. A proof-of-concept prototype system simulating the order entry, partner search and selection, and contracting in a virtual enterprise creation scenario is implemented to demonstrate the dynamic workflow definition and execution for inter-enterprise collaboration.

Keywords: Software agents; Web services; Workflow; Virtual enterprise; Enterprise collaboration

1. Introduction

Workflow management systems provide the automation of business processes where a collection of tasks is organized between participants according to a set of defined rules to accomplish some business goals (WFMC). The traditional inter-enterprise workflow is related to EDI (Electronic Data Interchange) and recently ebXML (electronic business XML) (ebXML), which coordinates business partners with pre-defined terms and thus constructs a close and secure business community. However, due to the lack of flexible mechanisms to deal with real-life situations, such as fast changing customer requirements and enterprise goal shifts, a static workflow definition designed at build time is inflexible to meet the complex, dynamic situations that happen at run time such as partner search and selection, bid negotiation, and workflow re-configuration.

In a dynamic, loosely coupled collaborative environment such as B2B or B2C, a flexible and adaptive inter-enterprise workflow is more suitable for those small and medium Internet-based business enterprises (Yan, Mammar, & Shen, 2001). For a flexible workflow, only a rough, role-based workflow pattern is defined at the beginning. The initial workflow needs to be specified or pre-processed until such a workflow pattern can be described by a clear and executable workflow definition.

Web services (W3C) provide an industrial standard for deploying, publishing, discovering and invoking enterprise’s services. From its emergence, many specialists have predicted that Web services will revolutionize the distributed computing paradigm and it will make various kinds of e-business (e.g., inter-enterprise collaboration and ASP paradigms) a reality. BPEL4WS is such a workflow process language for providing a formal specification of business processes and business interaction protocols. It uses executable business process and abstract process to ensure that different business processes can understand one another in a Web services environment, and that they can realize a dynamic composition. However, to achieve the required
dynamic composition of a workflow, BPEL4WS still falls in short of building a direct logical relationship between an abstract process definition and an executable process definition, except developers manually define such a relationship. Ontology and reasoning mechanisms must be added on top of BPEL4WS to describe the relationships between workflow components, such as tasks, services and resources, for automatic reasoning and interpretation of a dynamic workflow process. Thanks to the emergence of Semantic Web research and standardizing efforts, OWL (Web Ontology Language) (W3C) becomes a promising ontology representation language that combines the description logic, formal logic and Web service standards.

On the other hand, as significant research work has proved that software agents can provide a flexible, reconfigurable and coordinated approach to enhance workflow management (Xu, Qiu, & Xu, 2003; Yan et al., 2001), the merging of agents with workflow brings a promising result. An agent-based workflow can be considered as a workflow process that is planned, performed, communicated, and coordinated in a multi-agent environment, in which the workflow is decomposed into multi-level collaborative tasks and each task represents a logical piece of work that contributes to the process.

This paper presents an agent-based workflow model by integrating software agents, Web services, and workflow ontology to support dynamic workflow definition and execution for inter-enterprise collaboration. The proposed workflow model should include both internal and external workflow processes for their strong coherence relations.

The paper is organized as follows: Section 2 introduces Web services, workflow ontology, and agent technologies in workflow coordination; Section 3 proposes an agent-based workflow model; Section 4 presents the basic definition of workflow ontology and its reasoning; Section 5 describes a virtual enterprise creation case study; Section 6 depicts the implemented proof-of-concept prototype; finally, Section 7 provides a conclusion and discusses our future work.

2. Web services, workflow ontology and agents in workflow coordination

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility thanks to the use of XML, and they can then be combined in a loosely coupled way in order to achieve complex operations (W3C). The basic components of Web services, such as SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) and UDDI (Universal Description Discovery and Integration protocol), solve the interoperability problems of heterogeneous applications, but they are still not sufficient for large-scale industrial applications because of the lack of abilities to construct inter-enterprise workflow management systems. Facing this challenge, many workflow-oriented Web services components, such as ebXML, BPEL4WS, OWL (W3C) and OWL-S (W3C), have been proposed on top of those basic components for automating business process management. Among these components, BPEL4WS is particularly notable for providing a formal specification of business processes and business interaction protocols. BPEL4WS uses executable business process and abstract process to ensure that different business processes can understand one another in a Web services environment. However, the challenge is that Web services are hooked up on the fly and it may be beyond human capabilities to analyze the required services and compose them manually using BPEL4WS. On the other hand, manually created WSDL files for a process workflow could not meet the requirements of dynamic virtual enterprise collaborations.

Workflow ontology is being widely used to describe the relationships between workflow components, such as tasks, services and resources, for automatic reasoning and interpretation of a dynamic workflow process. OWL (W3C) is a standard ontology representation language that combines the description logic, formal logic and Web standards. OWL-S (W3C) is an OWL ontology evolved from DAML-OIL for Web services (DAML). It describes a set of classes, properties and relations that are specific to the description of Web services. The OWL-S consists of three service models: the service profile model for describing service advertisements, the process model for the actual program that realizes the service, and the service grounding model for message transportation.

Since OWL-S provides semantic-based service description, service discovery and logic reasoning and BPEL4WS offers the automation of a business process, some researchers have tried to combine the OWL-S with BPEL4WS to use both of their merits (Mandell & McIlraith, 2003). For example, Liu, Khalaf, and Curbera (2004) built a direct mapping from DAML-S to BPEL4WS for composite processes and atomic processes.

Considering the dynamic instantiation of a flexible workflow, the runtime flexibility required by a business workflow can naturally be achieved when we consider merging the agent concept into workflow automation scenario. Software agents can provide a flexible, reconfigurable and coordinated approach to enhance workflow management at both inter- and intra-enterprise levels (Shen & Norrie, 2001; Shen, Norrie, & Barthes, 2001). A software agent system satisfies the requirement of inter-enterprise workflow management in that:

- agent-based workflow can be considered as a workflow process that is planned, performed, communicated, and coordinated in a multi-agent environment;
- the workflow is decomposed into multi-level collaborative tasks and each task represents a logical piece of work that contributes to a process;
• a software agent is an execution entity on behalf of its role, knowledge, or resource to perform workflow tasks;
• workflow is used to control the interactions between agents.

Some research efforts have been focused on the use of agents for the composition of Web service-based workflow (Blake, 2003; Korhonen, Pajunen, & Puustjarvi, 2003; Sycara, Paolucci, Soudry, & Srinivasan, 2004; Zeng, Ng, Bentallah, & O’Dell, 2001). Korhonen et al. (2003) presented a design of a planner agent-based Web service to automatically compose Web service-based workflow. Vieira, Casanova, and Ferrao (2004) introduced an ontology-driven architecture to provide a flexible workflow execution. This architecture supports a mechanism to handle presuppositions and choose alternative sub-workflows based on sub-workflow ontology definition. However, most of the above work is limited to Web service-based workflow composition. In fact, if we consider a workflow as the combination of internal workflow and external workflow (Zeng et al., 2001), their logic relation for dynamic workflow composition is still under investigation. Both Web services and agent technologies can be used for inter-enterprise workflow automation. In fact, the dynamic agent-based workflow model plus the Web service-based interoperable protocols can generate a flexible, reconfigurable and coordinated approach to fulfill business process management both across enterprises and within an enterprise.

The integration of software agents and Web services can be realized at both the design level and the implementation level. At the design level, we encapsulate Web services as agent models so that each agent functions on behalf of a Web service in its action and relation to the environment. In this sense, we can treat a Web service as a semi-autonomous agent. On the other hand, Web services can be used to describe the external behaviors of software agents. Therefore, agents can be used to build high-level models with flexible interaction patterns, while Web services are more suitable for solving the interoperability problem of various applications in real implementations. At the implementation level, UDDI, WSDL and SOAP provide such capacities as discovery, deployment and communication, while specifications such as BPEL4WS provide service composition and process enactment.

3. The proposed agent-based workflow model

A workflow model typically includes a set of concepts that describe processes, tasks, roles, and dependencies among tasks. The proposed agent-based workflow model consists of the following elements as:

- **Workflow**: a workflow is a partial or total ordering of a set of tasks.
- **Task**: a task is a partial or total sequence of operations, descriptions for human actions, or other tasks.
- **Resource**: various resources include documents, data records, images, phones, fax machines, printers, etc.
- **Role**: a role is a placeholder for a human skill or an information system service required to perform a particular task.
- **Agent**: agents are humans or information systems that fill roles, perform tasks, and interact during workflow execution.

In general, the agent-based workflow model takes advantages of agent coordination mechanisms. An agent coordination mechanism is a description of actors and roles in that an actor playing a particular role carries out a set of activities (Weiss, 1999). In fact, coordination model (Omicini, Zambonelli, Klush, & Tolksof, 2001) uses a coordinated problem-solving method to perform workflow management where a coordination agent performs the functions of planning, coordinating and managing, while users and resource agents execute the proposed tasks. To simplify the workflow ontology, we abstract the workflow process to an agent-based workflow model according to the layered role players. The main role player in the coordination model is called workflow coordination agent, which is a software agent responsible for coordinating the workflow process. The agent-based workflow model is proposed with four layers: workflow management layer, coordination layer, service layer, and resource layer, as shown in Fig. 1.

The workflow management layer contains workflow definition tools that are mainly used to create business process representations, including the definition of business processes, rules, and logics. A workflow engine is a software service that provides a run-time execution environment for workflow instances. It is capable of initiating utilities to activate appropriate applications for the execution of planned activities. The coordination layer is the second layer whose responsibility is to coordinate the activities at the inter-enterprise level. In this layer, a coordinator agent is required, whether it be a broker, a mediator or an ontology agent. The service layer is composed of a set of service agents that represent their physical counterparts or legacy applications. The service agents implement the executable workflow tasks and exchange messages between roles. In our model, service agents are supplier agents at inter-enterprise level and resource agents are task execution entities at the intra-enterprise level. A resource agent has knowledge about its environment and acts as what it is expected. At the resource layer, workflow management is implemented by legacy business systems, human operators, customers, machines, or other applications.

4. Workflow ontology and reasoning

Ontology includes a vocabulary for representing knowledge for some topics and a set of relationships and properties. By building up the workflow ontology, we can automatically reason and interpret workflow at run time...
with high flexibility and also take advantages of the reusability of existing program modules with minimum redesign and re-development efforts. A workflow generally has a hierarchical structure in that its tasks and services can be further composed to sub-classes, properties and relations.

In general, ontology generation can be summarized as three approaches: (1) bottom-up—from specification to generalization; (2) top-down—from generalization to specification; and (3) middle-out—from the most important concepts to generalization and specialization. Bottom-up approach is widely used when considering real world situations, because most workflow systems are not built from scratch but from existed systems. Legacy systems are required to be integrated in most cases.

We propose a three-level workflow ontology as shown in Fig. 2. Firstly, at the workflow ontology level, the Abstract Workflow ontology (AW) has two sub-classes: the Execution Workflow ontology (EW) and the Abstract Task ontology (AT). The reason we directly map an AW to an EW is not only to integrate existing workflow systems, such as ERP systems, but also to enhance the efficiency of implementation in some specific situations. Secondly, at the task ontology level, an abstract task is decomposed into a set of Execution Tasks (ET) and these tasks are further matched with Abstract Services (AS). Finally, at the service ontology level, the abstract service is provided by either the Internal
enterprises (IS) or the External Services (ES). The difference between an IS and an ES is that an IS is to be executed inside the software system or within the enterprise boundary; while an ES is executed by outside systems or other enterprises.

The core workflow ontology entities in the proposed model are defined as follows:

- **Abstract workflow** (AW): AW defines an abstract workflow ontology which describes abstract workflow processes. An AW can be an execution workflow or a collection of abstract tasks. However, for the former case, each particular AW can only be mapped to one such execution instance.

- **Execution workflow** (EW): EW is a concrete workflow ontology that defines a detailed workflow procedure. EW can be decomposed into a set of execution tasks without the definition of abstract tasks. Due to the elimination logic reasoning procedure, EW is efficient at execution but less flexible at run time configuration.

- **Abstract task** (AT): AT is an abstract task ontology entity that has the properties of task name, task type, task deadline, task pre- and post-conditions, task status, and task execution result. An abstract task represents a composite task that has at least one atomic task (execution task).

- **Execution task** (ET): An ET is an atomic task that can be matched with an abstract service. ET class not only inherits the properties and relations of AT, but also has its own properties such as task creator, task status (which may include initial state, processing state, or final state), operations, and results.

- **Abstract service** (AS): AS is a service root that defines an abstract service corresponding to an execution task. In fact, an AS can connect a task with concrete services (or resources) that perform this task. An abstract service can be accomplished by either of two kinds of concrete services: internal services (IS) and external services (ES).

- **External service** (ES): ES is on behalf of external services that include Web services or other external services. The part of ES ontology (Web service) can be extracted from the UDDI registry. The relation of OWL-S profile and UDDI has been clearly described in BPEL4WS.

- **Internal service** (IS): IS ontology represents a service that can perform a task or a part of the task. The IS can be an internal work unit, software, or resource.

OWL is a Web ontology language, in which the semantic meaning is defined and translated by descriptive logic (Zeng et al., 2001). OWL ontology can be described in the domain of classes (concepts), properties and relationships between classes and properties. In fact, our definition of workflow ontology uses the same syntax as OWL. When a workflow instance is implemented, the top-level workflow ontology is dynamically interpreted and decomposed into low-level ontology by logic reasoning. For example, we define a simple abstract order task which has two sequential tasks: Task1 and Task2. We use OWL to express this example as follows:

```xml
<owl:Class rdf:ID="Order">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Task1"/>
    <owl:Class rdf:about="#Task2"/>
  </owl:unionOf>
</owl:Class>

<owl:Class rdf:ID="Task2">
  <rdfs:subClassOf rdf:resource="#Order"/>
  <owl:intersectionOf rdf:parseType="owl:collection">
    <owl:Class rdf:about="#Task1"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#status"/>
      <owl:hasValue rdf:resource="#done"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="Task1">
  <rdfs:subClassOf rdf:resource="#Order"/>
  <rdfs:subClassOf rdf:resource="#done"/>
</owl:Class>
```

In this example, we use the union relation at abstract task class “order” to describe its two subtasks “task1” and “task2”. And also we define intersection and restriction condition to express the relation of the two tasks. In fact, OWL consists of a set of constructors and axioms which can be used to express the complex relations of workflow ontology. At workflow design time, when a designer defines a new workflow process, he only needs to consider the upper level workflow ontology, such as the abstract workflow and its abstract tasks, and defines properties and related abstract tasks of the abstract workflow. The designer can also directly define the abstract workflow ontology as an execution workflow and thus relax runtime interpretation burden on the workflow engine. At workflow run time, when a workflow instance is triggered, the whole workflow process is logically reasoned from high-level ontology and dynamically interpreted according to the status returned by abstract tasks and execution tasks. However, in some situations, human operations may be needed in process ontology interpretation especially when the definition of workflow ontology is imperfect or incomplete.

5. A case study: virtual enterprise creation process

A virtual enterprise is defined as a temporary alliance of enterprises that come together to share skills and resources in order to better respond to business opportunities, whose cooperation is supported by computer networks and
adequate IT tools and protocols (Camarinha-Matos & Afsarmanesh, 1999). The life-cycle of a virtual enterprise is generally considered to be a four-stage process: creation, operation, evolution and dissolution. Among them, the first step (virtual enterprise creation) involves dynamically generated partnerships and dynamically composed service workflow in order for the successful operation of a virtual enterprise.

We consider a virtual enterprise is dynamically created following a process of order generation, partner search and selection, bid negotiation and contract awarding. Workflow is used to define the business process logics that are shared by the participants of a formed virtual enterprise. If we consider the workflow definition as a “class” in programming, a virtual enterprise can be considered as a running instance of such a class which is triggered by customer requirements, created by its lifecycle, controlled by workflow management, executed by workflow engine and dismantled once its goal is fulfilled.

5.1. System architecture and components

As shown in Fig. 3, the agent-based system architecture includes four kinds of agents: Workflow planner agent, Broker agent, Ontology agent and Service agents. Workflow planner agent controls the workflow. Broker agent stands in the middle and takes care of task coordination, task meditation and communication with outside services. Ontology agent performs ontology-based reasoning and provides match-making services to the broker agent and service agents. Service agents are responsible for task execution at the lowest level. Such an architecture is supported by the three-level workflow ontology proposed in the previous section.

![Fig. 3. A case study: virtual enterprise creation process.](image)
When a customer order triggers the agent-based workflow, the broker agent selects one workflow pattern, queries the ontology agent for a potential partner list, and broadcasts a “Call-for-Bid” using a SOAP message to all potential suppliers. After the customer selects the order bids and makes contracts with suppliers, the workflow instance can be specified and implemented for operation management. At the enterprise level, each enterprise manages its own tasks and jobs for goals propagated from the inter-enterprise level. Therefore, a temporary link is built according to the workflow definition between chosen service agents and coordination agents at the inter-enterprise level.

At design time, ontology agent extracts the service ontology from the registration information on UDDI (bottom-up approach) while workflow ontology and task ontology are built on top of the extracted service ontology (top-down approach). At run time, the upper workflow ontology is decomposed and dynamically interpreted as the lower level service ontology. For example, the workflow planner controls the whole workflow process according to the task status returned; the broker agent queries the ontology agent to locate the service agents for obtaining the best solution.

Major components in Fig. 3 are described as follows:

**Web portal** is a Web interface for users to access and manage related information. From the Web portal, customers set their profiles, place orders, select bids, and confirm the contract information. Behind the Web portal, orders are received and processed by the broker agent.

**Workflow planner** is a software agent that is responsible for coordinating the workflow process. It is also responsible for maintaining the upper-level workflow ontology, including abstract workflow ontology, execution workflow ontology, abstract task ontology, and task ontology. The abstract workflow ontology and execution workflow ontology are defined at the workflow ontology repository while the abstract task ontology and the execution task ontology are defined at the task ontology repository. The workflow planner includes a workflow engine and a workflow ontology engine. The workflow engine is a software service that provides a run-time execution environment for workflow instances. It is capable of initiating utilities to activate appropriate applications for the execution of planned activities. The workflow ontology engine is an ontology-reasoning tool that performs the decomposition process from the upper workflow ontology to the lower workflow ontology. When a workflow process is triggered, the workflow planner selects an abstract workflow definition from the workflow ontology repository. According to the defined relationships and properties, the workflow ontology engine finds out an execution task from either an execution workflow or an abstract task. Then, the workflow

Fig. 4. A sample sequence diagram.
engine implements a workflow instance and passes the control of the execution task to the workflow planner agent.

Broker agent plays the roles of service discovery, task execution, task mediation and task coordination. For example, as shown in Fig. 3, when the customer triggers the workflow process through the Web portal, the broker agent will submit the query to the workflow planner and receive an execution task from the planner. Then the broker agent will query the ontology agent for a list of services that match with this task. For an internal service, the task will be directed to the corresponding internal service agent; otherwise, the broker agent will interact with the external service agents to execute the task. After task execution, results will be returned to the workflow planner. The difference between the broker agent and the workflow planner is that the former performs task coordination while the latter is responsible for workflow reasoning and execution.

Ontology agent provides semantic integration services, responds to service queries and performs ontology reasoning and match-making. In order for the ontology agent to compare the requested capabilities with the advertised capabilities, requested and advertised capabilities must be formulated in a way that they are comparable in terms of service performance. The parameters and metrics of Quality of Services (QoS) also need to be defined at the UDDI for the formulation of services and the validation of service performance. A Web service ontology using OWL-S is defined where capabilities are annotated with a set of terminologies. In our system, the ontology agent can discover the WSDL descriptions by accessing the UDDI registry and retrieving service ontology descriptions through OWL-S profiles. When the broker agent submits a service query to the ontology agent, the ontology agent will check first its local service ontology repository then the UDDI registry, find the matching services, and return this matching list to the broker agent. An ontology agent may have advanced learning abilities in order to gather non-functional performance information of services on the fly, such as QoS (Quality of Service).

Service agent is an agent on behalf of a service entity that is capable of providing certain services. In an enterprise, there are usually a number of service agents. The coordination among these agents could be cooperative (when they unite for an order) or competitive (when they compete for an order). An external service agent registers its services at the UDDI registry and an internal service agent registers
at its ontology agent. The service agent will commit to a task only when the task is awarded to it by the broker agent.

**UDDI** is a static ontology repository that provides registration and look up services. UDDI can be implemented inside an enterprise and performs like a private UDDI, and it can also be implemented at a shared space for multiple enterprises for providing public information as public UDDI. Thus, one or more UDDI registries can be employed to provide enterprises with standard terms used in communication languages and knowledge related to these terms’ definitions, attributes, relationships and constraints. However, the search mechanism supported by UDDI is limited to keyword matches and it is critically in lack of semantic support. To enable more sophisticated matchmaking capacity, the OWL-S profile model provides a richer service description.

### 5.2. Major steps for creating a virtual enterprise

In this section, an online order scenario is envisaged to demonstrate the agent-based workflow coordination. As a common sense, a virtual enterprise is created in response to an online order placement from the user. Several major steps constitute the virtual enterprise creation process: service ordering, partner search, bidding, partner selection, and contracting. Detailed procedure of the aforementioned processes is shown as a sequence diagram in Fig. 4.

**Service ordering**: The user places an order through the Web portal. The submitted service order triggers the order workflow process. A service order process is separated as a sequence of order input, order generation and order submission. Once the order is submitted, for each individual order, the broker agent needs to query the workflow planner for task decomposition and workflow instantiation and the workflow planner agent makes its decision by workflow ontology reasoning from the abstract workflow definition to the executable workflow definition. When the broker agent gets an execution task, it tries to find a service agent to execute the task.

**Partner search**: The search and selection of business partners is a critical step in the formation of a virtual enterprise (Bremer, Mundim, Michilini, Siqueira, & Ortega, 1999; Hao, Shen, Wang, & Lang, 2003). Partner search can be done based on following resources: (1) private information resources such as the local service registry list of the broker agent; (2) ontology service registry at the ontology agent site; and (3) public service registry such as the public UDDI repository. In fact, the partner search process is carried out by the broker agent through checking sequentially these three resources with decreased priorities.

![Fig. 6. Screen shot of ordering process on Web portal.](http://localhost:8080/jsp-examples/jrc/servicebundle.jsp - Microsoft Internet Explorer)
The result of partner search step is a list of matching service agents that best serve the accomplishment of the identified task.

**Bidding process:** After the broker agent recognizes the potential service agents, it needs to negotiate with them for cost estimation, production planning, and conflict resolution so on. The CNP (Contract Net Protocol) (Smith, 1980), commonly used as an interaction protocol in a multi-agent community, is adopted for this inter-enterprise negotiation process. A “Call-for-Bid” message initialized by the broker agent and encoded using a SOAP message is sent to all potential service agents. Each service agent is required to return a “Bid” to the broker agent. The broker then collects the bids and feeds back the bid information to the customer.

**Partner selection:** The decision of partner selection is made by the broker agent (in the case of an automatic system) or the customer (in the case of an interactive system) after collecting all possible bids and checking with its decision rules. The most promising alternative for virtual enterprise partnership configuration is chosen based on the service time, cost, quality, trust, and other factors.

**Contracting:** Contracting process is used to achieve a business agreement between the customer and the service suppliers. A contract is generated by the broker agent mainly based on customer’s order, payment and shipping information as well as supplier’s bid information. The customer keeps the contract and its copies are sent to the selected service agents (participant enterprises) for execution.

6. Prototype implementation

A software prototype system is developed to demonstrate inter-enterprise collaboration in which an agent-based workflow model is combined with Web services...
and semantic workflow ontology. In fact, this system is not intended to simulate a complete business workflow process, but rather the dynamic evolving process of a workflow from its abstract definition to an executable implementation by a Contract Net-based negotiation mechanism.

In the implemented prototype system, the Web portal, broker agent, UDDI server are implemented on one computer and three supplier agents’ Web portals and Web services are running separately on three other computers. The relationships of the software components are briefly shown in Fig. 5.

The Web portals and Web services are developed using the Java Web Services Development Pack (JWSDP) (SUN. JWSDP). The JWSDP provides a convenient all-in-one package for developers who want to start building and deploying application Web services immediately. It is a superset of Java XML package. The main software tools used for system implementation include:

- Java APIs for Web services such as JAXP, JAXM, and JAXR.
- Tomcat as a Web portal test environment for Java Servlet or JSP.
- JSP tag library.
- JSSE (Java Secure Socket Extension) for testing Web service access via secure connections.
- Ant build tool for platform-independent build management.
- Java WSDL registry server which is a private UDDI server that can be deployed internally for service publication and discovery.

For simplicity of implementation, functions of the workflow planner agent are combined into the broker agent. The user places an order through a Web portal (Fig. 6). The broker agent, which is implemented also as a run-time workflow engine, has the functions of service discovery, coordination and mediation. The broker agent searches the ontology agent, negotiates with supplier agents, and provides supplier bids to the customer. However, the decision of bid selection and order contracting depends on the choice made by the customer in the current prototype.

After the customer selects the best order bid based on his knowledge and the award is accepted by the chosen supplier agent, a contract is generated automatically with the proposed cost, quality, and time schedule in the bid. The contract can be seen from the Web portal (Fig. 7) and each chosen supplier will receive a copy once it is confirmed.

As for service discovery, only the UDDI is implemented for ontology agent to search about. The interface of the UDDI server is shown in Fig. 8. Several organizations (supplier agents) together with services, descriptions, and service binding information are registered on the UDDI server.

![Fig. 8. Screen shot of registered services on UDDI server.](image)
Since the current implementation only deals with the partner search and selection, the Contract Net Protocol or its modified version is enough for the coordination among the broker agent and supplier agents. Other Web services standards such as BPEL4WS will be adopted for further interactions/collaborations between business partners. Moreover, software agents for intra-enterprise systems are simulated with the support of the Autonomous Agent Development Environment (AADE). AADE (Hao, Shen, & Zhang, 2005) is a FIPA (FIPA specifications) compliant engineering-oriented agent framework developed at the National Research Council Canada’s Integrated Manufacturing Technologies Institute and it has been used for the implementation of an agent-based dynamic manufacturing scheduling system at the enterprise level, where each machining tool is represented by one resource agent (machine agent). Details of the agent-based manufacturing resource scheduling system implemented using AADE platform are described in Wang, Shen, and Hao (2005).

7. Conclusion and future work

Today, workflow management systems often face the problem of runtime process changes and we cannot expect a pre-asserted design solution to handle all these changes (van der Aalst, 1999). Thus, a dynamic workflow should provide two meanings of flexibility: design composition and runtime composition. Using the agent-based Web service workflow model proposed in this paper, the abstract workflow definition can be interpreted dynamically at run time through service agents’ negotiation. Web service workflow ontology is evolved task by task depending on changing situations of workflow execution. Therefore, the effectiveness of workflow process can be improved by the dynamic interpretation of process ontology, the flexibility of process composition, and the reusability of process description.

In this paper, we also present a case study of inter-enterprise collaboration with a proof-of-concept prototype to simulate the order entry, partner search and selection, and contracting in a virtual enterprise creation scenario. Contract Net-based agent negotiation is the primary mechanism to assist the dynamic workflow instantiation and collaboration. The proposed approach is particularly suitable for integrating heterogeneous software and hardware systems existed within an enterprise or among collaborative enterprises. Our future research in this area would be in two directions: improving semantic integration approaches to address complex scenarios and developing practical applications with industrial partners.

References


Mandell, D. J., & McIlraith, S. A. (2003). Adapting BPEL4WS for the Semantic Web: The bottom-up approach to Web service interopera-


puter-Aided Engineering, 8(1), 17–30.


W3C. OWL-S. Available from http://www.w3.org/Submission/OWL-S/.


