Using Ontology-based BDI Agent to Dynamically Customize Workflow and Bind Semantic Web Service

Chih-Hao Liu and Jason Jen-Yen Chen
Dept. of Computer Science and Information Engineering
National Central University, Jhong-Li, Taiwan
Email: 945402024@cc.ncu.edu.tw, jasonjychen@gmail.com

Abstract—**As the Web gradually evolves into the semantic web, the World Wide Web consortium (W3C) recommends that web ontology language (OWL) be used to encode semantic information content over the Web. Semantic web is an essential infrastructure to enhance Web to obtain better integration of information and intelligent use of web resources. Moreover, a web service is annotated by web ontology language for service (OWL-S) to form a semantic web service that, however, is a static description. The OWL-S based semantic web services thus are reactively invoked by users. How to dynamically coordinate, composite, or discover the services is an important issue.**

We use agent to proactively interpret the static OWL-S description. And, the Belief-Desire-Intention (BDI) model is applied to develop BDI agent. This work thus proposes a ontology-based BDI agent architecture, in which a BDI agent dynamically generates customized workflow, and binds semantic web services. The architecture includes four parts: 1) Application Ontology, which is description of a specialized domain, 2) Operation ontology, which is description of BDI agent, 3) Ontology-based BDI agent engines, which interpret corresponding operational ontology to dynamically generate workflows, and 4) Java agent development environment extension (JadeX) platform that our architecture is based on. Through JadeX, our BDI agent can dynamically bind semantic web services according to customized workflows.

Index Terms—- **Ontology, BDI Agent, Workflow, Semantic Web Service.**

I. Introduction

In traditional web, the objective of a web site is to provide information for user. As web gradually evolves into web 2.0, users not only share information to constitute social group, such as Blog, but also integrate distributed information from different sources. The web service, described by Web Service Definition Language (WSDL), is a popular approach to provide service on the Web. Along with WSDL, the Simple Object Access Protocol (SOAP) and Unified Discovery and Definition Identify (UDDI) are suggested by the World Wide Web consortium (W3C) to publish, access, and deploy services on the Web. Furthermore, web services are published by various service providers in different domains and are statically existent in distributed environment. Thus, how to coordinate, composite, or discover services is an important issue [1, 34, 35].

We also see that Web gradually evolves into the semantic web. The W3C recommends that web ontology language (OWL) be used to encode semantic information content over the Web [2, 3]. The OWL is a formal language to define domain knowledge to generate an OWL document called domain ontology, which defines significant terms and their relationships within a specific domain, such as software engineering [39, 40]. Moreover, the ontology, called operational ontology, specifies the operational concepts as terms, and their relationships as property. A software entity can execute a task according to this ontology. In particular, the Web Ontology Language for Service (OWL-S) describes the semantics for web service to generate semantic web services [22]. Unfortunately, the OWL-S based semantic web services are static descriptions that are reactively invoked by users.

On the contrary, an agent is a program that proactively cooperates with other agents to execute a complex task, such as making appointment or scheduling, through a sequence of communication acts [4, 5, 46]. And, the foundation for intelligent physical agents (FIPA) is a popular standard to define 22 communicative acts to regulate the format of communication among agents [15-21]. Moreover, the Belief-Desire-Intention (BDI) model is a well-known approach to provide intelligent actions according to the state of environment [23]. The BDI model is applied to the development of a kind of agent called BDI agent [43, 45]. "Belief" means what an agent believes; "desire" represents a goal that user wants to achieve; and "intention" is a plan that can be executed to satisfy the goal. To achieve the user's request (goal), an agent selects the actions (plans) according to what it believes. Some researchers argue that a web service is like an action [24]. On the other world, when an agent has to execute an action, it selects suitable web services to match the action [41, 42]. Further, an agent also communicates with other agents to coordinate the selected web services to achieve the user's request [7]. However, a goal to be achieved generates a lot of actions, which form different workflows for various users. And, a particular user follows a customized workflow to dynamically bind web services.

This work proposes a ontology-based BDI agent architecture, in which a BDI agent proactively binds web services according to a customized workflow that is dynamically generated to meet a user request. Notably, the user ontology is defined to provide user's preference that includes both static and dynamic information.

II. RELATED WORK

The OWL-S annotates three kinds of semantics to a web service to generate semantic web service: 1) service profile, 2) service grounding, and 3) service process [22]. A service provider usually publishes unrelated service profiles for user to browse. However, the user cannot easily use the profiles to compose web services [6, 7, 8]. As the annotations are static descriptions, the autonomous and social agents seem more suitable to compose the services [7, 9, 33]. Further, workflow is a good approach to describe the problem solution for a user [36, 37, 38]. Also, it is a static description as well. Sebastian et al. proposed the semantic-based BDI agent to coordinate the workflows [29]. However, it lacks the knowledge representation of the workflow. This work thus proposes a BDI agent to generate the customized workflow to dynamically compose semantic web services.

Ontology is a formal language that defines knowledge as resources, which can be easily shared over the web. Further, the ontology is capable of integrating heterogeneous information easily and provides the standard interface for software entity to access [10, 31, 32]. Daud et al. developed an approach for learning a semantic representation underlying a user's interests using his/her personal search history [11]. Xing Jian et al. used a user model to provide customized information services [12]. On the web, customization is important as it provides suitable services to different users [13, 14, 30]. In our architecture, we define operational and application ontology that assist the integration of agent and semantic web services.

The Java agent development framework (JADE) is a popular platform to develop a FIFA-compatible agent [25]. And, the workflows and agents development environment (WADE) is a software platform based on Jade that provides support for the execution of task defined according to the workflow metaphor [26]. However, JADE does not support to develop a BDI agent. The JadeX extends JADE to support the BDI model through describing BDI in extensible markup language (XML) format and through developing plan as Java class [27, 44]. Further, the JadeX process project also supports business process modeling notation (BPMN) and goal-oriented process modeling notation (GPMN) [28]. However, the workflows above are all statically designed in advance, and do not support dynamic and customized workflow.

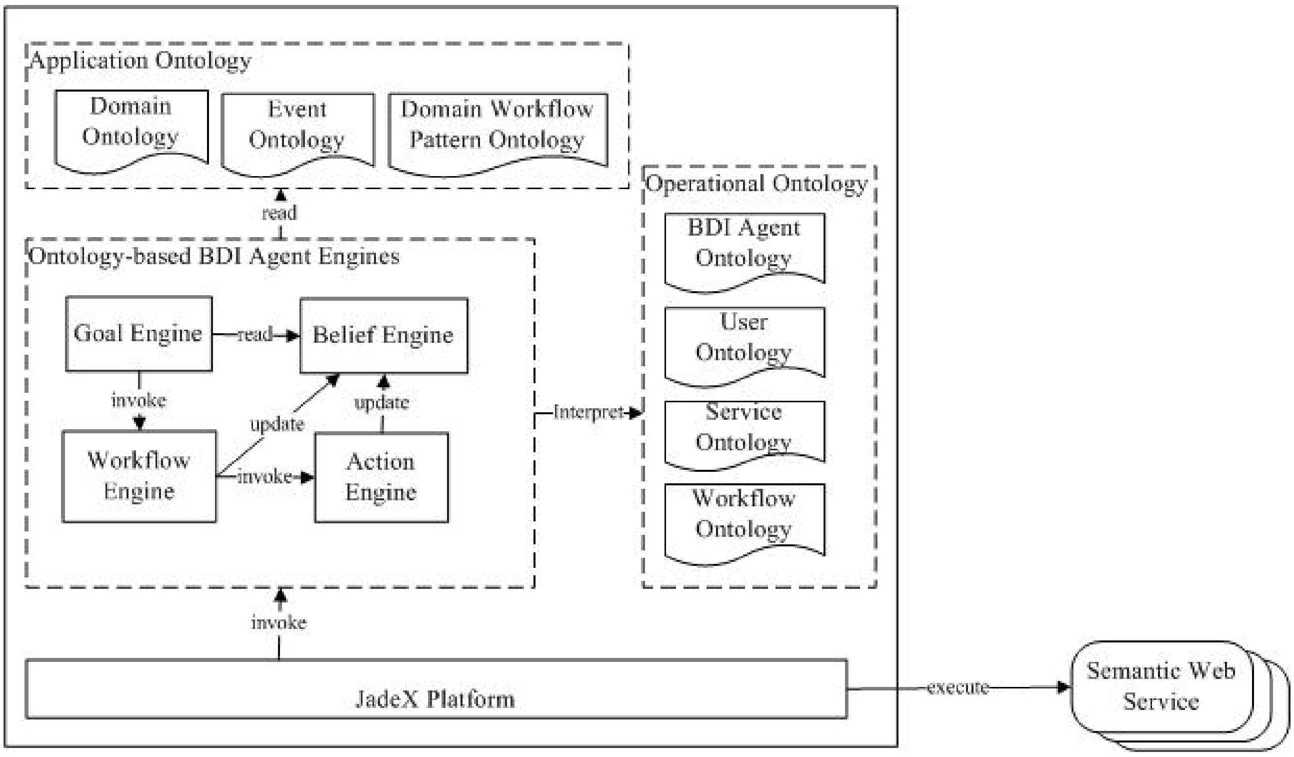

Figure 1: The Ontology-based BDI Agent Architecture

III. ONTOLOGY-BASED BDI AGENT ARCHITECTURE

This work proposes a ontology-based BDI agent architecture, which has four main parts: 1) application ontology, 2) ontology-based BDI agent engines, 3) operational ontology, and 4) JadeX platform (see Figure 1).

Our architecture is based on JadeX platform, in which the BDI agent handles user's request (described as a goal), which will be converted to a workflow according to the beliefs (user ontology) of the agent. Further, the BDI agent proactively binds web services according to the workflow. And, the application ontology has: 1) domain ontology, 2) event ontology, and 3) domain workflow pattern ontology. The domain ontology describes the terms of a specialized domain provided by an expert. The event ontology illustrates an event that is triggered by a condition in the domain. Finally, the domain workflow pattern ontology defines a workflow about a task in the domain. The BDI agent engines and the operational ontology are described next.

A. Ontology-based BDI Agent Engine

The ontology-based BDI agent engines contain four engines: 1) goal engine, 2) belief engine, 3) workflow engine, and 4) action engine. First, the goal engine processes user's request to form a goal, which has three types: 1) achieve goal, 2) execute goal and 3) maintain goal (to be described). Second, the belief engine handles the user ontology to create agent's beliefs. Third, the workflow engine generates a customized workflow according to the user ontology and user's goal. Last, the action engine decides which semantic web services will be matched to a workflow.

Figure 2 shows the process of ontology-based BDI agent engines. The goal engine has two parts: 1) user handler and 2) goal generator. The former is a listener to receive user's request. The latter generates a goal description according to BDI agent ontology. Further, the internal of workflow engine has two parts: 1) goal handler and 2) workflow generator. The former processes the goal description received from the goal generator. The latter generates the workflow description according to workflow and domain workflow pattern ontology. For example, a user wants to buy a book and the goal description about buying book will be received by the goal handler. After that, the workflow generator will read domain workflow pattern ontology to get the standard workflow about buying book, which contains four basic actions: 1) search book, 2) recommend book, 3) order book, and 4) payment. Assume that a user just wants to buy a book called "Fundamental of Data Structure". The workflow generator then reads user ontology and goal description to generate the customized workflow, which contains three actions: 1) search book, 2) order book, and 3) payment. Notably, the "recommend book" is not included in this customized workflow.

Moreover, the belief engine has two parts: 1) belief updater and 2) belief translator. The former updates the post-condition of customized workflow and semantic web services that user agent invokes. The latter reads user ontology to create beliefs that stand for a user agent's belief. Finally, the action engine has two parts: 1) workflow handler and 2) semantic web service match maker. The former pre-processes a workflow description received from goal generator. The latter binds semantic web services according to the workflow description and user ontology.

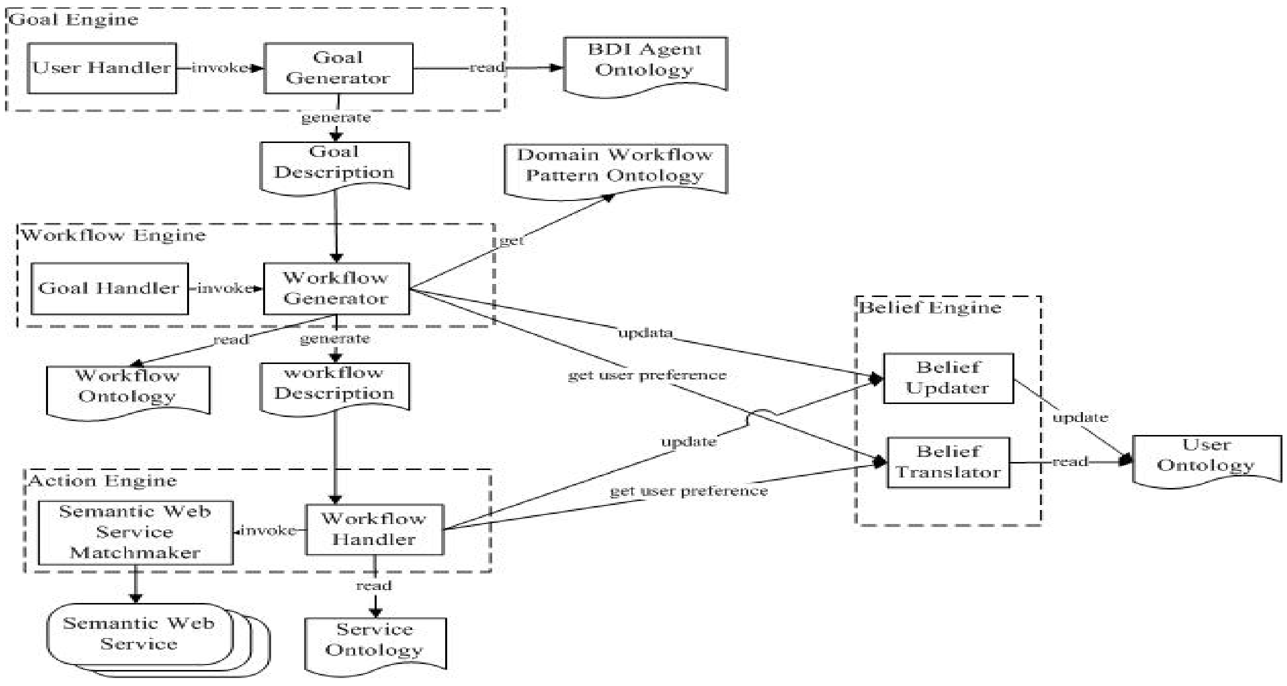

Figure 2: The Process of Ontology-based BDI Agent Engines

B. *Operational Ontology*

The operational ontology includes: 1) user ontology, 2) BDI agent ontology, 3) service ontology, and 4) workflow ontology. The user ontology expresses a user's preferences. The BDI agent ontology describes a BDI agent that includes what it believes, what it should achieve, and what it must do (Figure 3). The service ontology provides service pools for registration of web services. The workflow ontology defines elements of workflow, such as sequence or iteration.

a. *BDI Agent Ontology*

This section depicts the high-level concepts of BDI agent ontology (see Figure 3), which consist of: 1) the goal that the agent wants to achieve, 2) the belief that it believes in, and 3) the plan that it will execute. The three parts are respectively defined as ”Goal”, “Belief” , and “Workflow” classes. Together, the three classes compose an “Agent” class. The ”Goal” class contains three sub-classes: ”AchieveGoal” , “PerformGoal”, and “Maintain Goal”. “AchieveGoal” specifies the condition that must be satisfied after executing a user task. ”PerformGoal” simply specifies executing a user task. And, ”MaintainGoal” specifies the condition that must be satisfied at every time interval.

The ”Belief” class specifies what facts an agent believes. And, the facts are defined as “Facts” class, which contains a sub-class ”Fact_Statement” that is represented in a XML literal statement. Further, the ”Fact_Statement” class has two subclasses: 1) ”Condition_Statement” and 2) “ValidTime_Statement” class. The “Condition_Statement” class specifies a fact is true when the condition is satisfied. And the condition is defined as ”Condition” class. The “ValidTime_Statement” class specifies that a fact is true given a valid time, defined as ”ValidTime” class that is described in “xds:datatime” type. The ”ValidInstant” and “ValidPeriod” classes are subclasses of the “ValidTime” and represent a time instant and time interval, respectively. The “Workflow” class stands for a workflow that is dynamically generated when a user executing a task. A workflow entity, which is defined as “WorkflowEntity” class, is a basic element of a workflow. The detailed of workflow ontology will be described next.

The main concept of definition of BDI agent ontology is to dynamically generate a customized workflow according to various users when a user agent executing tasks. Moreover, a customized workflow dynamically binds semantic web services according to user ontology. This provides intelligent and dynamic web services.

b. *Workflow Ontology*

This section shows the workflow ontology that defines a workflow as a “Workflow” class (see Figure 4), which has basic element “Workflow Entity” class that has two sub-classes: 1) atom workflow entity, and 2) functional workflow entity. The atom workflow entity has three subclasses: 1) start entity, 2) end entity, and 3) activity entity. The start entity, defined as “Start Entity”, stands for the beginning of a workflow. The end entity is defined as “End Entity” that stands for the end of a workflow. The “Activity Entity” corresponds to a plan defined as “Plan” class, or a service pool defined as “Service Pool” class. The “Plan” class specifies an action that an agent executes. For example, a user agent executes a web service. The “Service Pool” class specifies an abstract service that contains a lot of references referring to various web services. Further, a service pool is classified by service ontology for service providers to publish web services to it.

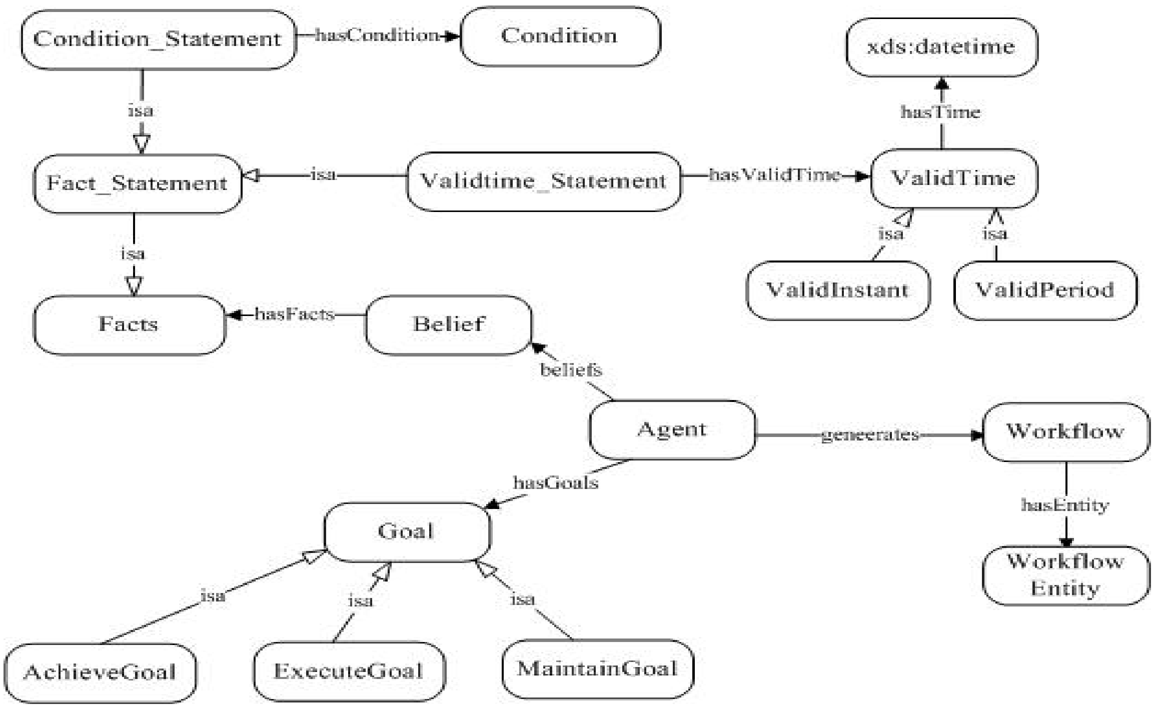

Figure 3. The BDI Agent Ontology

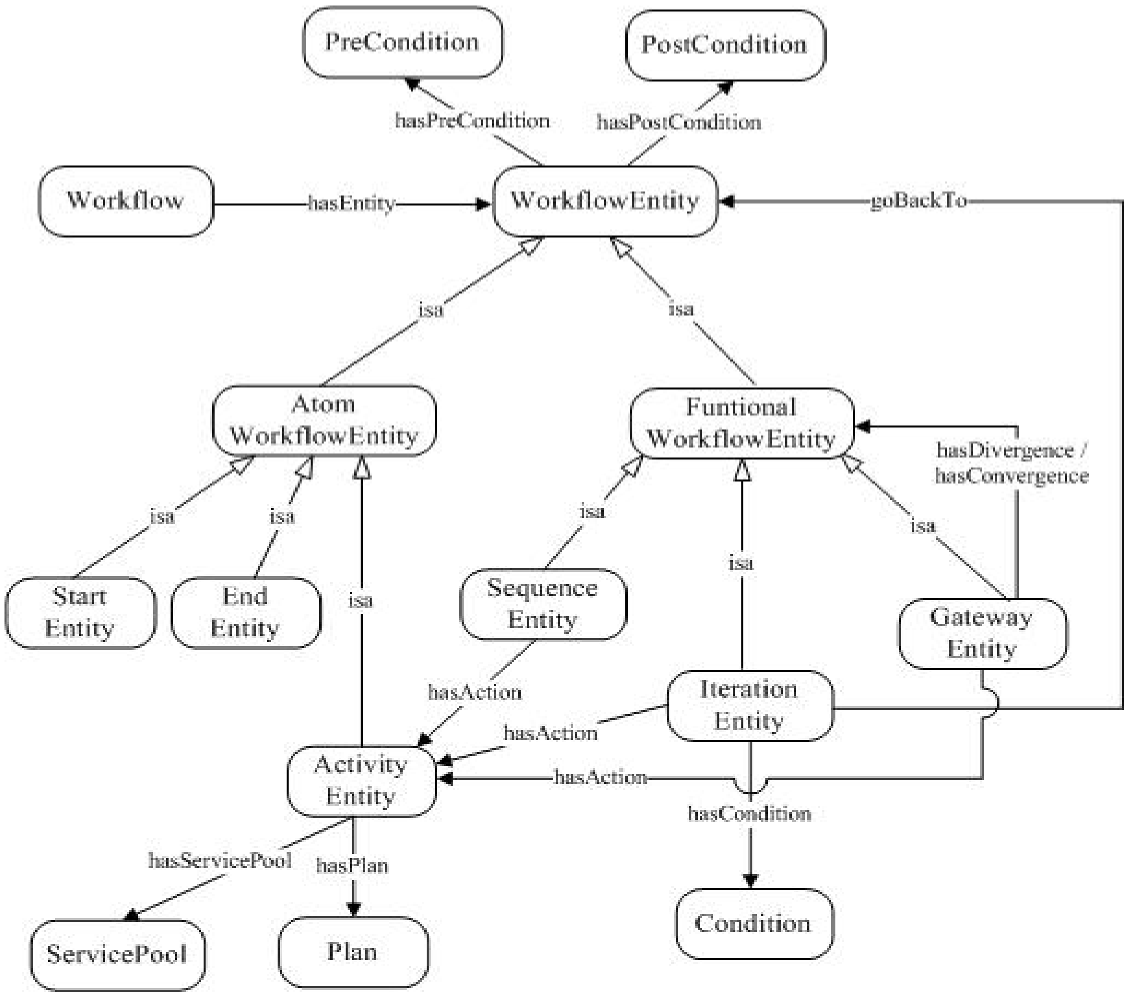

Figure 4 The Workflow Ontology

The functional workflow entity has three subclasses: 1) sequence entity, 2) iteration entity, and 3) gateway entity. The sequence entity has an action, which is defined as "Activity Entity" class. The iteration entity has a condition, which is defined as the "Condition" class that specifies a condition. When it is satisfied, a workflow will jump to another workflow entity. The gateway entity is defined as "GatewayEntity" class, which is used to control divergence and convergence of workflow. Moreover, the workflow has pre-condition and post-condition, which are respectively defined as "PreCondition" class and "PostCondition" class. The pre-condition describes a condition that must be satisfied before executing the workflow. The post-condition describes a condition that is satisfied after executing the workflow.

c. *User Ontology*

This section shows the user ontology that records user's preferences, which include dynamic workflow and web services information. The user ontology about workflow is shown in figure 5.

In figure 5, the user preference is defined as "UserPreference" class, which has workflow statement defined as "WorkflowStatement" class. A workflow statement describes that an agent, which is an actor, executes a domain workflow that is defined as "DomainWorkflow" class and belongs to a web site. Further, the domain workflow has workflow entities, which are classified into three categories: 1) executed entity, 2) optional entity, and 3) failed entity. The executed entity is defined as "ExecutedEntity" class, which records executed workflow entities of the domain workflow. The operational entity is defined as "OptionalEntity" class, which presents the workflow entities of the domain workflow that have been executed by an agent, but the execution is not a necessity to the completion of the workflow. The failed entity is defined as "FailEntity" class, which records the workflow entities of the domain workflow that causes failure.

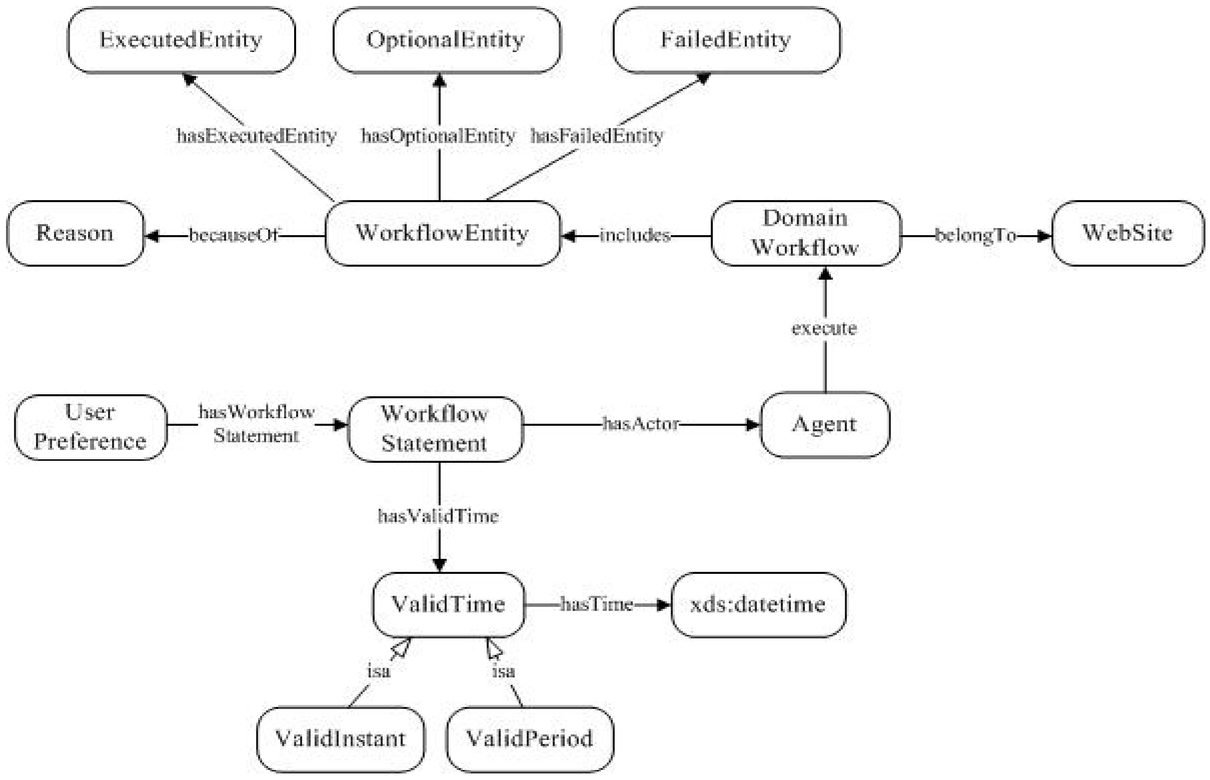

Figure 5 User Ontology about Workflow

Moreover, each workflow entity has a reason, which is defined as "Reason" class, for storing the condition and reasoning about selecting optional entity and failed entity. The workflow statement has a valid time to record the time instant that an agent finishes executing this domain workflow. Next, the service information about user ontology is shown in figure 6.

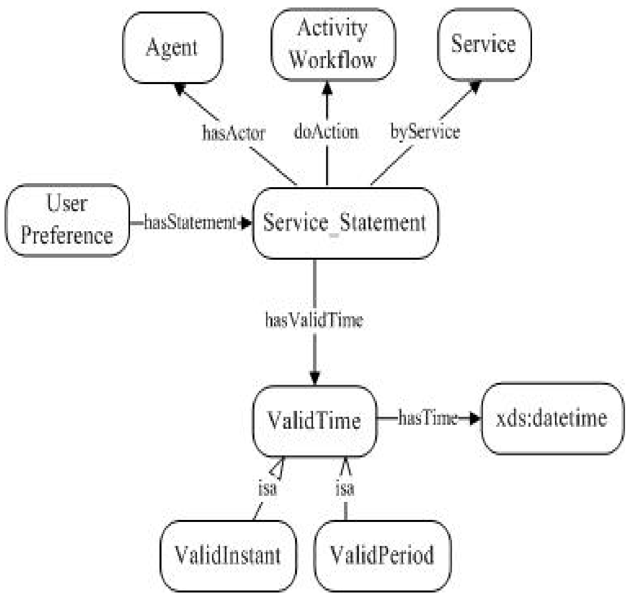

Figure 6 User ontology about Service

In figure 6, the user preference has the service statement, which is defined as "ServiceStatement" class. A service statement describes that an agent executes an activity workflow entity through the services. Again, it also has a valid time instant to record when an agent finishes executing the web services.

d. Servicer Ontology

This section illustrates the service ontology that describes web service information for an agent as shown in figure 7.

In figure 7, an agent manages some web services, which are defined as "Service" class. The "Publisher" class records the service's publisher. The "ServiceProfile" class points to the profile of the OWL-S service. The "Domain" class defines the domain the service belongs to. The "Value" describes the popular value of the service and is evaluated by the popular formula as shown below:

$$V(s, t+1) = V(s, t)*\delta^{-b} + NoUser\ (s)_{t=month} + Fre\ (s)_{t=month}$$

where V(s, t+1) is the popular formula, s is a web service, t_{i+1} is a new time at which a user uses this service, and the δ^{-b} is a decay function. The number of user ($NoUser(s)_{t=month}$) is the number of users using this service in one month. The $Fre(s)_{t=month}$ represents the usage frequency of the web service in one month.

In addition, the "Constraint" class defines the constraints about using the web service. The "Description" class expresses some information about the web service for user to understand.

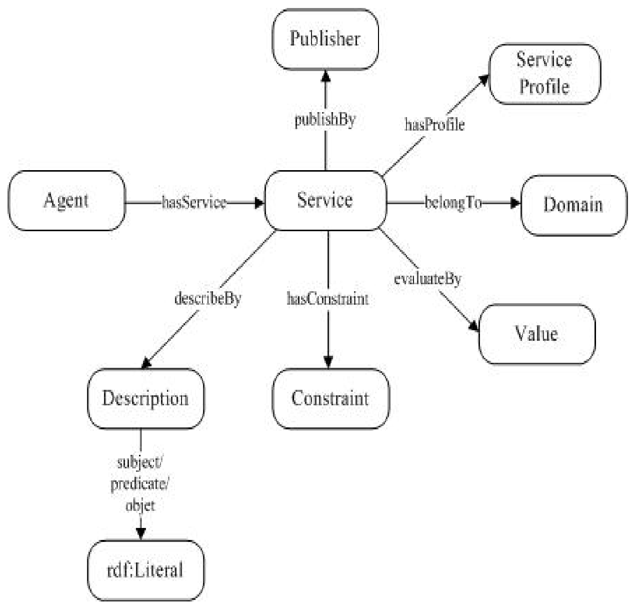

Figure 7 Service Ontology

IV. TWO EXAMPLES

This section shows two users of online book buying examples to illustrate dynamically generating a customized workflow to meet user request (Figure 9). The simple online book buying workflow is shown in figure 8.

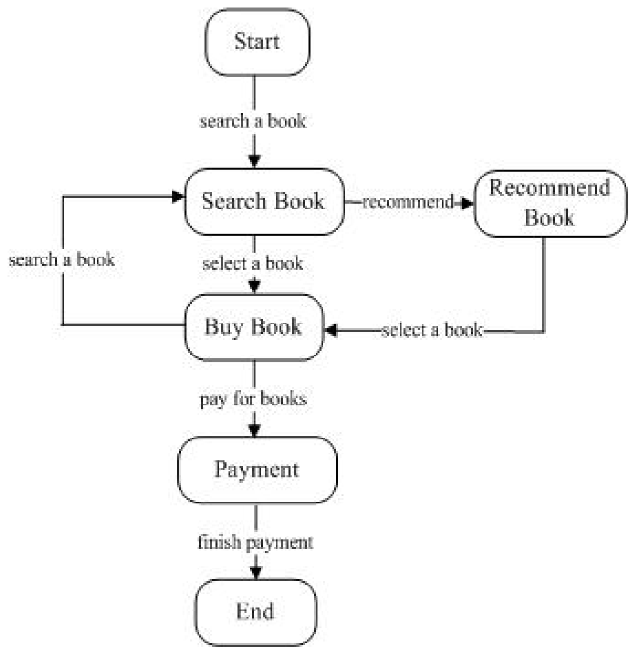

Figure 8 the simple online book buying workflow

In figure 8, there are six workflow entities defined to form the simple online book buying workflow. And, each user will follow this workflow to buy books. The beginning of online book buying workflow is the start entity. First, a user will do the "search book" entity to search book he/she wants to buy, and then he/she selects a book. After that, the system records the book into book-list and goes to next workflow entity "Buy Book". Of course, the user could select another book until he/she wants to pay for all the books he/she has selected. The system thus could go to "Search Book" entity or to "Payment" entity. Notably, a user selects a book by searching the book or recommending the book according to user presences. After payment, the online book buying workflow is finished and the system goes to "End" entity.

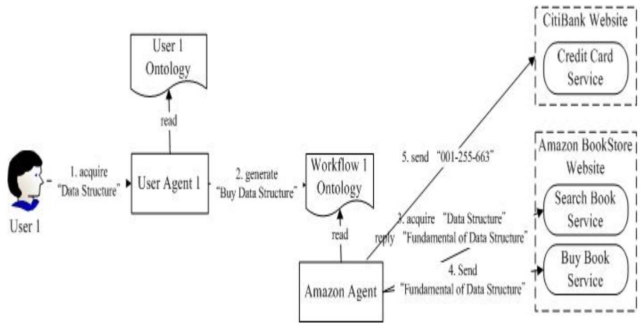

Figure 9(a): User 1 buys book from Amazing Bookstore

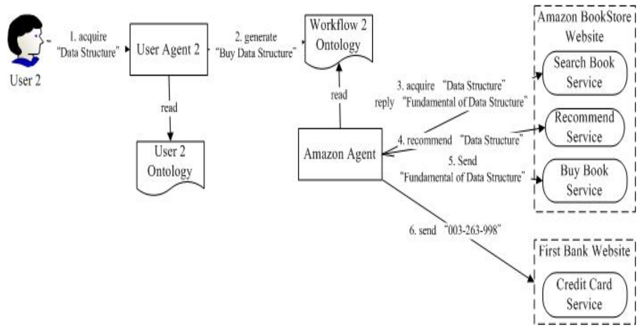

Figure 9(b): User 2 buys book from Amazing Bookstore

Amazing book store website provides a "search book" service for user. When a user wants to buy a data structure book, he/she can delegate a user agent to send "acquire Data Structure". The user agent then invoke goal engine to read BDI agent ontology to generate a goal description. After that, the user agent invokes belief engine to get what the user agent believes. And, the belief engine will read user ontology to get user's information. The user agent then will invoke workflow engine to generate the customized workflow, which is based on domain workflow patter ontology and user ontology. Finally, the user agent will send the customized workflow to the Amazon agent that will binds semantic web services to the customized workflow according to user ontology to meet user request. Next, we will show how two customized workflows can be generated in Figrue 9(a) and 9 (b) respectively.

Figure 9(a) shows that the user 1 wants to buy a data structure book and he/she delegates the user agent 1 to send "acquire Data Structure". After that the user agent 1 reads the user agent 1 ontology to get user information, and then generates “buy data structure workflow” (workflow 1 ontology). Finally, the user agent 1 sends the workflow to Amazon agent. When Amazon agent receives the workflow, it will bind the various semantic web services according to the workflow as shown in Figure 10(a). First, it invokes “search book” service to obtain information of data structure book. After that, it invokes “buy book” service to order a data structure book. Finally, it gets user 1’s credit card number “001-255-663” and then invokes “credit card” service that provided by CitiBank web site.

```
<B0:BDI Agent rdf:ID=”User 1”>
  …
 <W0:StratEntity rdf: ID=”start 1”>
   <W0:toBeginEntity rdf:resource=”#SearchBook 1”>
 </W0:StartEntity>
 <W0:GatewayEntity rdf: ID=”SearchBook 1”>
   …
   <W0:has Action>
   <W0:ActivityEntity rdf:ID=”SearchBookAct”>
     <W0:hasPlan>
       <W0:Plan rdf:ID=”SearchBookActP1”>
       http://AmazonWebSite/SearchBookService
       </W0:Plan>
     </W0:hasPlan>
   </W0:ActivityEntity>
   </W0:hasAction>
 </W0:GatewayEntity>
 <W0:IterationEntity rdf:ID=”BuyBook 1”>
   …
   <W0:goBackTo rdf:resource=”#SearchBook 1”/>
   <W0:toNextEntity rdf:resource=”#Sequence 1”/>
   <W0:has Action>
     <W0:ActivityEntity rdf:ID=”BuyBookAct”>
        <W0:hasPlan>
          <W0:Plan rdf:ID=”BuyBookActP1”>
            http://AmazonWebSite/BuyBookService
          </W0:Plan>
        </hasPlan>
     </W0:ActivityEntity>
   </W0:has Action>
   …
 </W0:IterationEntity>
 <W0:SequenceEntity rdf:ID=”Sequence 1”>
   <W0:hasAction>
     <W0:ActivityEntity rdf:ID=”Payment 1”>
       <W0:hasPlan>
         <W0:Plan rdf:ID=”PaymentActP1”>
          http://CitiBankWebSite/CreditCardService
         </W0:Plan>
       </W0:hasPlan>
     <W0:/ActivityEntity>
     <W0:toNextEntity rdf:resource=”#End1”>
   </W0:has Action>
 </W0:SequenceEntity>
 <W0:End rdf:ID=”End1”/>
 …
</B0:BDI Agent>
```

Figure 10 (a) The segment of Workflow 1

```
<B0:BDI Agent rdf:ID=”User 2”>
  …
 <W0:StratEntity rdf: ID=”start 2”>
   <W0:toBeginEntity rdf:resource=”#SearchBook 2”>
 </W0:StartEntity>
 <W0:GatewayEntity rdf: ID=”SearchBook 2”>
   …
   <W0:has Action>
   <W0:ActivityEntity rdf:ID=”SearchBookAct”>
     <W0:hasPlan>
       <W0:Plan rdf:ID=”SearchBookActP2”>
       http://AmazonWebSite/SearchBookService
       </W0:Plan>
     </W0:hasPlan>
   </W0:ActivityEntity>
   </W0:hasAction>
   <W0:has Action>
   <W0:ActivityEntity rdf:ID=”RecommendBook”>
    <W0:hasPlan>
     <W0:Plan rdf:ID=”RecommendBookAct”>
     http://AmazonWebSite/RecommendBookService
     </W0:hasPlan>
    </W0:hasPlan>
   </W0:AcitivtyEntity>
 …
 </W0:GatewayEntity>
 <W0:IterationEntity rdf:ID=”BuyBook 2”>
   …
   <W0:goBackTo rdf:resource=”#SearchBook 2”/>
   <W0:toNextEntity rdf:resource=”#Sequence 2”/>
   <W0:has Action>
     <W0:ActivityEntity rdf:ID=”BuyBookAct”>
        <W0:hasPlan>
          <W0:Plan rdf:ID=”BuyBookActP2”>
            http://AmazonWebSite/BuyBookService
          </W0:Plan>
        </hasPlan>
     </W0:ActivityEntity>
   </W0:has Action>
   …
 </W0:IterationEntity>
 <W0:SequenceEntity rdf:ID=”Sequence 2”>
   <W0:hasAction>
     <W0:ActivityEntity rdf:ID=”Payment 2”>
       <W0:hasPlan>
         <W0:Plan rdf:ID=”PaymentActP2”>
          http://FirstBankWebSite/CreditCardService
         </W0:Plan>
       </W0:hasPlan>
     <W0:/ActivityEntity>
     <W0:toNextEntity rdf:resource=”#End2”>
   </W0:has Action>
 </W0:SequenceEntity>
 <W0:End rdf:ID=”End2”/>
 …
</B0:BDI Agent>
```

Figure 10(b) The segment of Workflow 2

Figure 9(b) shows that the user 2 also wants to buy a data structure book. And the user agent 2 also sends “buy data structure workflow” (“workflow 2 ontology”) to Amazon agent. However, the user 2 wants to obtain recommendation information about data structure. Thus, the Amazon agent will invoke “recommend” service after invoking “search book” service as shown in Fig. 10(b). After that, it will also invoke “buy book” service to order a data structure book. Finally, it gets user 2’s credit card number “003-263-998” to and then invokes “credit card” service that provided by FirstBank web site.

V. CONCLUSIONS

This work proposes a ontology-based BDI agent architecture, which includes four parts: 1) Application Ontology, which is description of a specialized domain, 2) Operation ontology, which is description of BDI agent, 3) Ontology-based BDI agent engines, which interpret corresponding operational ontology to dynamically generate workflows, and 4) Java agent development environment extension (JadeX) platform that our architecture is based on. We expect the advantages are as below:

1. The workflow ontology provides the knowledge representation to describe the domain workflow. The agent with BDI ontology dynamically generates customized workflow for users according to the user ontology.

2. The user ontology includes user’s usage information, which contains two parts: 1) workflow and 2) web service. Through JadeX, our BDI agent can dynamically bind semantic web services according to customized workflows.

ACKNOWLEDGMENT

The authors wish to thank the reviewers of the WASE International Conference on Information Engineering. This work was supported in part by a grant from NSC99-2221-E-008-084.

REFERENCES

[1] Jiehan Zhou, Juha-Pekka Koivisto, and Eila Niemela, “A survey on Semantic Web Service and a Case Study”, in the Proceeding of the 10th International Conference on Computer Supported Cooperative Work in Design, 2006

[2] OWL Web Ontology Language Overview, W3C Recommendation, available at http://www.w3.org/TR/owl-features, 10 February 2004.

[3] Ding, L., Kolari, P., Ding, Z., Avancha, S., & Finin, T., Using Ontologies in the Semantic Web: A Survey. UMBC eBiquity Publication, October 2005.

[4] J. Hendler, “Agents and the Semantic Web”, Intelligent systems, vol. 16, Issue 2, pp. 30-37, Mar-Apr, 2001.

[5] J, Yen, X. Fan, S. Sun, T. Hanratty, and J. Dumer, “Agents with shared mental models for enhancing team decision makings”, Decision Support Systems, vol. 41, pp. 634-653, 2006.

[6] W.T. Tsai, Jay Elstion, Yinong Chen, Composing Highly Reliable Service-Oriented Application Adaptively, The Fourth IEEE International Symposium on Service-Oriented System Engineering, pages. 118-122, Jhong-Li, Taiwan, R.O.C., 18-19 December 2008.

[7] C.H. Liu, Y.F. Lin, and J.Y. Chen, “Using Agent to Coordinate Web Service,” the International Journal of Computer Science and Information Security (IJCSIS), vol. 2, no. 1, pp. 18-25, 2009.

[8] Maximilien, E.M. and Singh, M.P., “A Framework and ontology for dynamic Web services selection”, Internet Computing, IEEE Volume 8, Issue 5, Sept.-Oct. 2004 Page(s): 84 – 93

[9] Francisco, G. S., Rafael V. G., Rodrigo, M. B., Leonardo, C., and Jesualdo T. F. B., “An ontology, intelligent agent-based framework for the provision of semantic web services”, Expert Systems with Applications, 36(2): 3167-3187, March 2009.

[10] A. Pease, I. Niles, and J. Li, The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications, In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web, Edmonton, Canada, Jul.-Aug. 2002.

[11] Daoud, M., Tamine, L., Boughanem, M., Chebaro, B., “Learning Implicit User Interestes Using Ontology and Search Histor for Personalization,” In International Web Information System Engineering – International Workshop on Personalized Access to Web Information (WISE=PAWI 2007), Nancy, France, 2007

[12] Xing Jian and Ah-Hwee Tan, “Learning and Inferencing in User Ontology for Personalized Semantic Web Search”, Information Sciences, pages 2794-2808, 2009.

[13] R.L. Roberto, S.R.P. da Silva, “An approach for identification of user’s intentions during the navigation in semantic websites”, The Semantic Web: Research and Applications, 4th European Semantic Web Conference, 2007, pp. 371–383.

[14] A. Micarelli, F. Gasparetti, F. Sciarrone, S. Gauch, “Personalized search on the world wide web”, P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Methods and Strategies of Web Personalization, 2007, pp. 195–230.

[15] Foundation for Intelligent Physical Agents, FIPA Contract Net Interaction Protocol Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00029/SC00029H.pdf.

[16] Foundation for Intelligent Physical Agents, FIPA Iterated Contract Net Interaction Protocol Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00030/SC00030H.pdf.

[17] Foundation for Intelligent Physical Agents, FIPA Request Interaction Protocol Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00026/SC00026H.pdf.

[18] Foundation for Intelligent Physical Agents, FIPA Request When Interaction Protocol Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00028/SC00028H.pdf.

[19] Foundation for Intelligent Physical Agents, FIPA Brokering Interaction Protocol Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00033/SC00033H.pdf.

[20] Foundation for Intelligent Physical Agents, FIPA Recruiting Interaction Protocol Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00034/SC00034H.pdf.

[21] Foundation for Intelligent Physical Agents, FIPA Communicative Act Library Specification, Dec. 6, 2002, available at: http://www.fipa.org/specs/fipa00037/SC00037J.pdf.

[22] OWL-S: Semantic Markup for Web Services, November 22, 2004, available at: http://www.w3.org/Submission/OWL-S/
[23] Birgit Burmeister, M. Arnold, Felicia Copaciu and Giovanni Rimassa, "BDI-agents for Agile Goal-oriented Business Processes", Proceedings of the 7th international joint conference on autonomous agents and multi-agent system: industrial track (AAMAS 08), 2008, pp. 37-44.
[24] P.A. Buhler, J.M. Vidal., "Semantic Web Service as Agent Behaviors", Agentcities: Challenges in Open Agent Environments, Springer-Verlag, 2003.
[25] JADE Java Agent DEvelopment Framework, available at http://jade.tilab.com/, May 2003.
[26] WADE Workflows and Agents Development Environment Tutorial, 04, February, 2009, available at http://jade.tilab.com/wade/doc/tutorial/WADE-Tutorial.pdf
[27] JADEX User Guide, 1, June, 2007, available at http://jadex-agents.informatik.uni-hamburg.de/docs/jadex-0.96x/userguide/userguide.pdf
[28] BPMN Business Process Modeling Notation Tutorial, 22, Dec. 2010, available at http://jadex-processes.informatik.uni-hamburg.de/xwiki/bin/export/BPMN+Tutorial/BPMN+Tutorial
[29] Sebastian Richly, Sandro Schmidt, and Uwe Assmann, "A Semantic-BDI-Based Approach to Realize Cooperative, Reflexive Workflow", Proceedings of the 8th world congress on Intelligent Control and Automation, July 6-9, 2010, Jian, China, pp. 1680-1685.
[30] Hao Liu, Ben Salem, and Matthias Rauterberg, "A Survey on User Profile Modeling for Personalized Service Delivery Systems", In Proceeding of IADIS International Conference on Interfaces and Human Computer Interaction 2009, pp.45-51.
[31] Samuil Nikolov and Anatoliy Antonov, "Framework for Building Ontology-based Dynamic Applications", Proceedings of the 11th International Conference on Computer Systems and Technologies – CompSysTech'10, pp. 83-88.
[32] Sheng-Yuan Yang, "A New Ontology-Supported Interface Agent", TENCON 2007 IEEE Region 10 Conference, Taipei, Taiwan, pp. 1-4.
[33] Hao Yang, Junliang Chen, Xiangwu Meng, and Ying Zhang,"A Dynamic Agent-based Web Service Invocation Infrastructure", ACHI'08 Proceedings of the First International Conference on Advances in Computer-Human Interaction, pp. 206-211
[34] E. Sirin, J. Hendler, and B. Parsia, "Semi-Automatic Composition of Web Service using Semantic Descriptions", in Web Service: Modeling, Architecture and Infrastructure Workshop in ICEIS 2003, April, pp. 17-24.
[35] A. Brogi, S. Corfini, and R. Popescu, "Semantics-based Composition-Oriented Discovery of Web Services", ACM Trans. Interet Technol., Vol. 8, no. 4, pp. 1-39, 2008.
[36] Francisco Burbera, Matthew Duftler, Rania Khalaf, and Douglas Lovell, "Bite: Workflow Composition for the Web", ICSOC 2007 International Conference of Service-Oriented Computing, Volume 4749, pp. 94-106.
[37] Khalid Belhajjame, Suzanne M. Embury, Norman W. Paton, Robert Stevens, and Carole A. Goble, "Automatic Annotation of Web Services based on Workflow Definitions", ACM Transactions on the Web (TWEB), Volume 2, Issue, 2, April 2008, Article 11
[38] Qiang He, Jun Yan, Hai Jin, and Yun Yang, "Adaptation of Web Service Composition Based on Workflow Patterns", ICSOC 2008 International Conference of Service-Oriented Computing, Volume 5364, pp. 22-37.
[39] F. Correa da Silva, W. Vasoncelos, D. Robertson, V. Brilhante, A. De Melo, M. Finger, and J. Agusti, "On the Insufficiency of Ontologies: Problems in Knowledge Sharing and Alternative Solutions", Knowledge Based Systems, 15(3):147-167, 2002.
[40] Villanueva-Rosales N and Dumontier M, "Describing Chemical Functional Groups in OWL-DL for the Classification of Chemical Compounds", in OWL: Experiences and Directions (OWLED 2007), co-located with European Semantic Web Conference (ESWC 2007), Innsbruck, Austria.
[41] Matskin M, Kungas P, Rao J, Sampson J, Peterson SA. "Enabling Web Services Composition with Software Agents", Proceedings of the ninth IASTED International Conference on Interest and Multimedia Systems and Applications (IMSA 2005), 2005 August 15-17, Honolulu, Hawaii, USA.
[42] Maamar Z, Mostefaoui SK, Yahyaoui H, "Toward an Agent-based and Context-oriented Approach for Web Services Composition", IEEE Transaction of Knowledge Data Engineering, 2005, 17(5), pp. 686-697.
[43] Liu Yong, Pu Shuzhen, Cheng Daijie and Cao Zehan, "Belief Characteristic's Research of BDI Model", Journal of Computer Research and Development, 2005, 42(1), pp.54-59.
[44] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI Reasoning Engine", In Bordini et al. [5], chapter 6, pages 149-174
[45] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf, "Goal Representation for BDI Agent Systems", In Proceedings of the Second Workshop on Programmin Multiagent Systems (Pro-, MAS04), 2004, pp. 44-65.
[46] Eric Platon, Marco Mamei, Nicolas Sabouret, Shinichi Honiden, and H. Van Dyke Parunak, "Mechanisms for Environments in Multi-agent Systems: Survey and Opportunities", Autonomous Agents and Multi-Agent Systems, 2007, Volume 14, Number 1, pp. 31-47.

Chih-Hao Liu was born in 1980, Liu received his Master degree of Information Engineering from Chao-Yang University of Technology in Taichung, Taiwan. He is currently a PhD candidate of the Nation Central University in Taiwan. He joined the Software Engineering laboratory in 2005 and researched on the Semantic Web and Agent. He has participated in several projects about semantic web and agent such as Service-oriented Information Market-place (SIM) project from 2005 to 2007, Agent Authoring Environment project from 2008 to 2009, and Dynamic Composition of Web Service Based on Semantic Agent Framework project from 2009 to 2010.

Jason Jen-Yen Chen is with the Department of Computer Science and Information Engineering in National Central University in Taiwan. He earned international recognition by winning Top, Third, and Fifth Scholar in the world in the field of System and Software Engineering in 1995, 1996, and 1997, respectively. The ranking is based on cumulative publication of six leading journals in that field. His current research interests include agile method and agent technology. He established a web site to advocate agile method called “Agile Method Nursery”, which is a pioneering initiative in Taiwan. He also organized an Agile Method Symposium in Taiwan. His vision is to help transform the hardware-centered computer industry in Taiwan into a software-centered industry.