An Open Web Service based Multi-Agent Framework for Context Aware Computing

Peng Chen[1], Junzhong Gu[1*], Rong Tan[1] and Zili Zhou[2]

[1]Department of Computer Science and Technology, East China Normal University
[2]College of Physics and Engineering, Qufu Normal University

[12]{pchen, rtan, zlzhou}@ica.stc.sh.cn;[*] jzgu@cs.ecnu.edu.cn

Abstract

With the development of context aware computing, an adaptive, reflective, scalable, dynamic reconfigurable framework is urgently required. However, most current frameworks fail to meet new challenges in scalability and dynamic reconfiguration. These can be mainly attributed to two issues. First, a framework's real hope for widespread adoption lies in its easy incorporation of existing third party resources. Second, the framework is required to better support the dynamic reconfiguration, which means framework behaviors should be altered according to the change of contexts. To address these issues, an Open Web Service based Multi-Agent framework (OWS-MA) is proposed and implemented. We illustrate the overall framework, elaborate on the individual goals of each part, and illuminate how each part responses the challenges. Our exploration discloses a promising way to the future developments of context aware application.

Keywords: *Context Awareness; Multi-Agent Framework; Open Web Service*

1. Introduction

Nowadays, enormous IOT (Internet of Things), mobile and context aware applications are emerging, such that context aware computing is focused by researchers. The research to find a suitable framework for context aware applications is urgently required. In [1], it analyzes the key features that an adequate framework should satisfy are adaptivity, reflectivity, scalability, dynamic reconfiguration and context awareness.

Adaptivity: applications should run efficiently and predictably in more situations. Monitoring of resource supply and demand should be executed; meanwhile adaptation and notification should also be made accordingly. Hence, framework should be designed to flexibly sense context and automatically optimize resource utilization.

Reflection: as summarized in [1], the demand of reflection is three-fold, (1) the runtime inspection and adaptation of the underlay implementation should be supported (2) for monitoring or altering the internal behavior, a simple yet useful mechanism should be offered to insert behaviors into the framework, and (3) the application should be able to inspect and adapt the framework behavior according to its needs.

Scalability: for the widespread of any framework, it should be able to integrate a large number of third party resources, such as services, sensors, and data sources. For services, the Web Service is the most dominance technology right now. It helps millions of applications implemented cross different platforms. For sensors and data sources, exposing unified Web Service interfaces is rational and feasible.

Dynamic Reconfiguration: Since contexts are changing all the time, the framework behavior should be changed accordingly. Furthermore, dynamic changes of the framework behavior will inevitably involve in resource re-evaluations, reallocations, and notifications to framework components to adapt to the changes.

Context Awareness: A trend of computing is to endow software the ability of automatically sensing the surrounding context. The contexts concerned in this paper include device characteristics, user activities and locations, available bandwidths and services, and other resources of the framework.

In this paper, we incorporate multi-agent technology to make our framework adaptive, reflective and dynamic configurable. An extension of Web Service, Open Web Service, is used to broaden the scalability of our proposed framework and cover the defects of Web Service technology. The main features of proposed framework are adaptivity, reflection, scalability, dynamic reconfiguration, and context awareness.

The rest of this paper is organized as follows. In Section 2, a brief introduction to related researches is given. In Section 3, the detail of the OWS-MA framework is illustrated. A prototype implementation is reported in Section 4. Finally, the last section presents the conclusions of this paper.

2. Related Work

2.1. Multi-agent Technology

From a development prospective, a context aware computing application involves a large number of distributed objects on networks. A promising solution facilitating the development of applications involving highly distributed objects is the multi-agent technology. There, each agent is a software entity which functions in a computable node with autonomy. The multi-agent technology masks out the problems of heterogeneity, facilitates transparent and robust distributed communications, and exposes a unified invoking interface. Intuitively, the multi-agent technology provides capabilities that are perfectly in-line with the above requirements of framework for context aware computing.

The unanimous uses of the multi-agent technology in new emerging applications, such as Dalica [2], ScudWare [3], Open Smart Classroom [4], also prove that the multi-agent technology is inherently in accordance with the requirements of context aware computing. Based on the aforementioned analysis, a multi-agent framework is proposed and implemented in this paper.

2.2. Web Service

As the development of internet technology, more and more applications become internet-based. This makes Web Service a quite reasonable communication approach for internet-based applications [1]. A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network [5]. It offers an approach to compose a single application of services implemented on different platforms. Since it is widely used among innumerous applications, the specific communication protocols used in DCOM, OpenCOM [6], CORBA [7], as well as OpenORB [8] become less preferable.

2.3. New Challenges in Context Aware Computing

Most existing frameworks such as DCOM, OpenCOM, CORBA, OpenORB, cannot response new challenges emerging in context aware computing in terms of scalability and dynamic reconfiguration. Since they are limited on a certain platform, such as Windows or CORBA, and result in lack of scalability.

The simple incorporation of Web Service technology does expand the scalability of proposed framework. While in context aware computing, stateful service and privacy protection are the two important issues. To remedy Web Service's statelessness and lack of privacy protection, the extension of Web Service mechanism is demanded.

For dynamic reconfiguration, the mere usage of multi-agent technology is not enough. A more flexible mechanism is required to guarantee the dynamic substitution of underlay services won't interferes upper applications.

To face these challenges, the Open Web Service and Application Constructor are proposed to remedy the inherent defects of Web Service technology and achieve dynamic reconfiguration, respectively. To respond the aforementioned challenges, the OWS-MA framework is proposed. In the following sections, we will illuminate how each part addresses the challenges.

3. The OWS-MA Framework

3.1. Agent

In this paper, the same with [1], an agent is defined as a software component, which executes specific tasks on behalf of someone (a person) or something (an organization or another agent) with some autonomy. The general structure of an agent implemented in our framework is depicted in Figure 1.

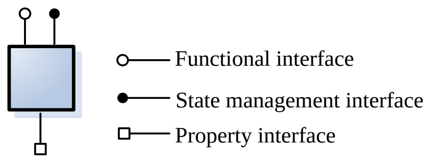


Figure 1. The General Structure of an Agent

For each agent, three interfaces are exposed, a functional interface to offer services, a state management interface to change the run-time internal status, and a property interface which provides an approach to achieve the reflectivity. Through the state management interface, application developers can create, suspend, or kill a service provided by each agent.

The agent technology solves the heterogeneity problem and exposes unified Open Web Service interfaces (detailed in Section 3.3) for application developers. The usage of agent technology makes our proposed framework more adaptive and reflective.

3.2. The Classification of Agents

Since the functionalities of agents are various, a classification is required for designing a multi-agent framework. From a functionality point of view, agents are organized into categories, as depicted in Figure 2.

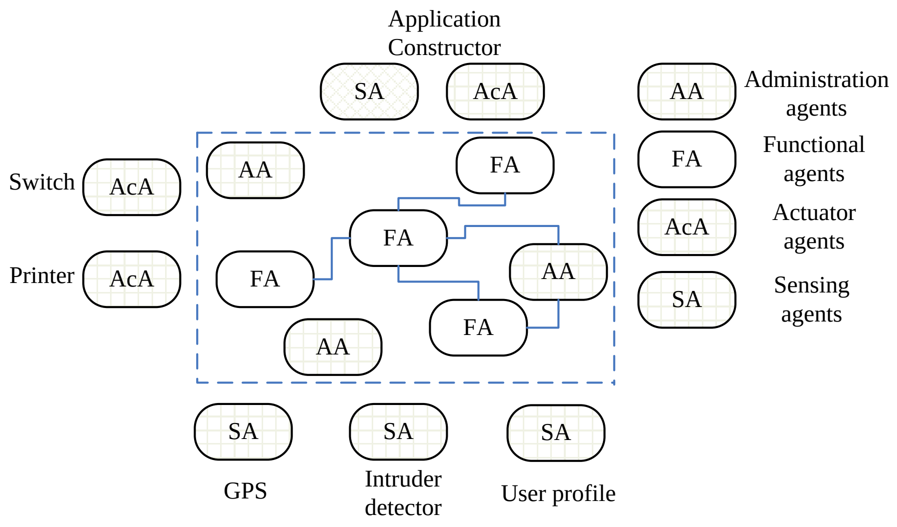


Figure 2. The Classification of Agents

Agents are roughly classified into administration agents, functional agents, sensing agents, and actuator agents, respectively. The descriptions and examples of each class are summarized in Table 1.

Table 1. The Classification of Agents

name	description	examples
Administration agents	Manage, monitor, audit the whole framework	State Query Agent, Directory Facilitator, Authority Manager.
Functional agents	Offer specific services to facilitate the development of application	Filtering agent, Fusion agent, Reasoning agent.
Sensing agents	Sense resources and information for the framework	GPS, Gravity sensor, Intruder detector, Web crawler.
Actuator agents	Perform reactions to the surrounding environment	Printer, Switch.

Through classification of agents, common features between agents are extracted to define agent templates. Then all the agent templates are organized in a hierarchy. Instead of starting from scratch, application developers can inherit from those agent templates and define their own agents.

3.3. Open Web Service

The statelessness, insecurity, and lack of privacy consideration confine the application of Web Service technology in context aware computing. In this paper, an extension of Web Service, Open Web Service, is proposed to cover these defects and used as a soft bus connecting each participating agents in applications throughout our solution.

In Open Web Service, the stateful Web Service, role based privacy protection, and authentication are implemented via encapsulating information into the header part of SOAP package; the request and response still remain in the body part. The states of Web service include active, suspend, processing, waiting. The communication security protection is fulfilled through the encryption of the SOAP package. The Open Web Service gives our framework an advantage in term of scalability when comparing with COM or CORBA.

3.4. The OWS-MA Framework

After preparing all the ingredients, the OWS-MA framework is shown in Figure 3.

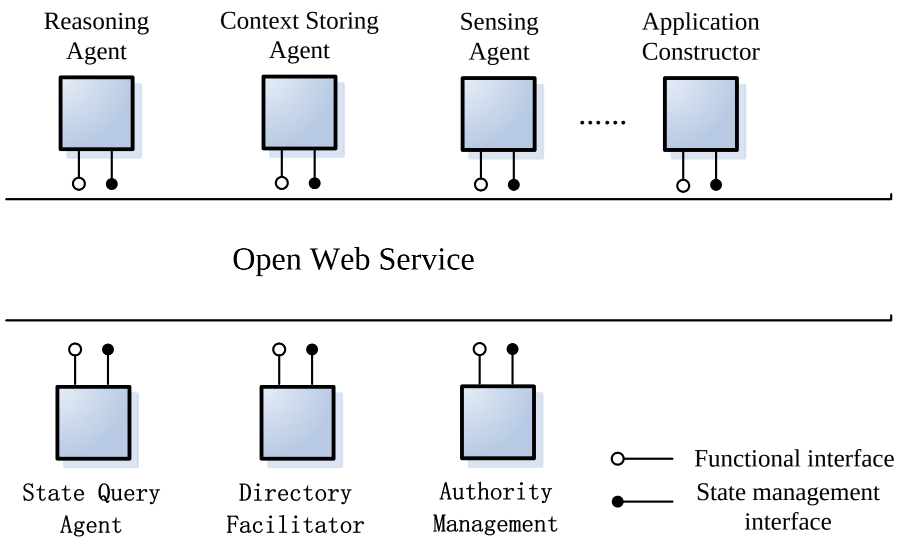


Figure 3. The Structure of the Proposed Multi-agent Framework

The Sensing, Reasoning, and Context Storing Agent are in charge of sensing, reasoning, and storing context respectively. The Application Constructor composes existing services into user specified applications. The State Query Agent offers a tool for application developers to monitor the state of each running agent. The Directory Facilitator serves as a directory to new agents discoveries. The Authority Manager stores the user-defined rules for authentications between agents and those between users and applications. Furthermore, it also exchange public key between communicating agents.

Our proposed framework releases application developers from bothersome agent distributions problem, and facilitates the development of distributed applications. The incorporation of multi-agent technology makes our proposed framework the ability of adaptivity, reflection and dynamic configuration.

To summarize, the advantages of combing agent technology with Open Web Service are fourfold, (1) it makes our proposed framework more adaptive and reflective, (2) it facilitates application developments, (3) it further expand the scalability of the proposed framework, and (4) it enhances the security and privacy protection.

4. Prototype Implementation

Our proposed multi-agent framework is implemented in an experimental system named GaCam [1]. GaCam, as a successor of LaMOC [9, 10], is a middleware system to support the construction and running of context aware applications, launched since 2009. In this section, several key components of the proposed framework are illustrated.

4.1. Agent Templates

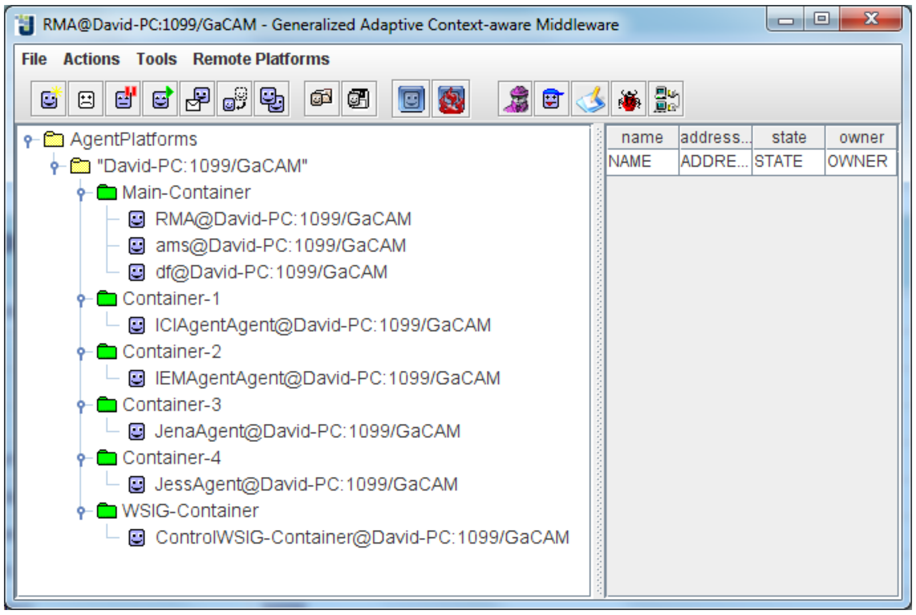
Some general agent templates are provided for application developers to create their own functional agents more conveniently. A location template is shown as follows:

```
Location extends SensingAgent{    / /such as GPS, intruder detector, web crawler.
    //Attributes{
        Longitude: real;
        Latitude: real;
        …
    }
    // Behaviors{
        // inheritable methods
            getContext();
            convertContext();
            filterContext();
            fuseContext();
        …
        //private methods
            getLocation();
    }
}
```

The properties of each agent are described by corresponding attributes, and the dynamic behavious of the agent are abstracted as methods. Methods can be divided into either inheritable or private base on the agent hierarchy. They can also be divided into functional or administrative according to their functionalities. In this template, some generic behavious, such as getContext(), convertContext(), filterContext and fuseContext() are presented. Agent templates facilitate the creation of agents, since it is not necessary for application developers to create their own agents from scratch.

4.2. Agent Administrator

As shown in Figure 4, agent administrator offers a tool to monitor and manually change the state of any agent in a single computable node. One can activate, suspend, kill any agent or dynamically alter its parameters. The life circle management of a local agent is also implemented in agent administrator.

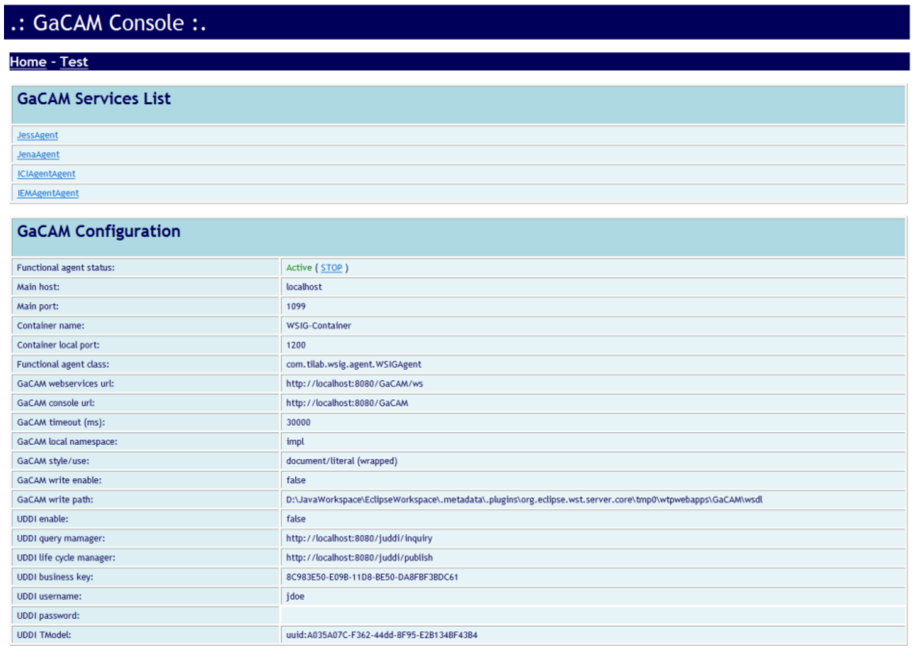

RMA@David-PC:1099/GaCAM - Generalized Adaptive Context-aware Middleware
File
Actions
Tools
Remote Platforms

```
AgentPlatforms
  "David-PC:1099/GaCAM"
    Main-Container
      RMA@David-PC:1099/GaCAM
      ams@David-PC:1099/GaCAM
      df@David-PC:1099/GaCAM
    Container-1
      ICIAgentAgent@David-PC:1099/GaCAM
    Container-2
      IEMAgentAgent@David-PC:1099/GaCAM
    Container-3
      JenaAgent@David-PC:1099/GaCAM
    Container-4
      JessAgent@David-PC:1099/GaCAM
    WSIG-Container
      ControlWSIG-Container@David-PC:1099/GaCAM
```

name	address...	state	owner
NAME	ADDRE...	STATE	OWNER

Figure 4. The GUI of Agent Administrator

The functional interface is exposed in an Open Web Service manner, as shown in Figure 5. Information, such as service names, URLs, offered services are described in according web pages.



GaCAM Services List

JessAgent
JenaAgent
ICIAgentAgent
IEMAgentAgent

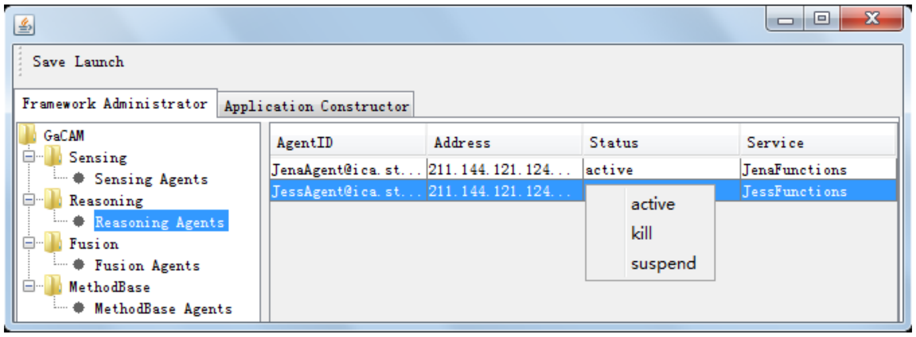
GaCAM Configuration

Functional agent status:	Active (STOP)
Main host:	localhost
Main port:	1099
Container name:	WSIG-Container
Container local port:	1200
Functional agent class:	com.tilab.wsig.agent.WSIGAgent
GaCAM webservices url:	http://localhost:8080/GaCAM/ws
GaCAM console url:	http://localhost:8080/GaCAM
GaCAM timeout (ms):	30000
GaCAM local namespace:	impl
GaCAM style/use:	document/literal (wrapped)
GaCAM write enable:	false
GaCAM write path:	D:\JavaWorkspace\EclipseWorkspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\GaCAM\wsdl
UDDI enable:	false
UDDI query mamager:	http://localhost:8080/juddi/inquiry
UDDI life cycle manager:	http://localhost:8080/juddi/publish
UDDI business key:	8C983E50-E09B-11D8-BE50-DA8FBF3BDC61
UDDI username:	jdoe
UDDI password:	
UDDI TModel:	uuid:A035A07C-F362-44dd-8F95-E2B134BF43B4

Figure 5. The Web Service Interface for an Agent

4.3. Framework Administrator

As shown in Figure 6, the framework administrator is in charge to monitor the status of all agents, both local and remote, of the applications. It also has the ability to activate, suspend or kill any local or remote agent. This tool also offers a way to manually reconfigure the whole framework.

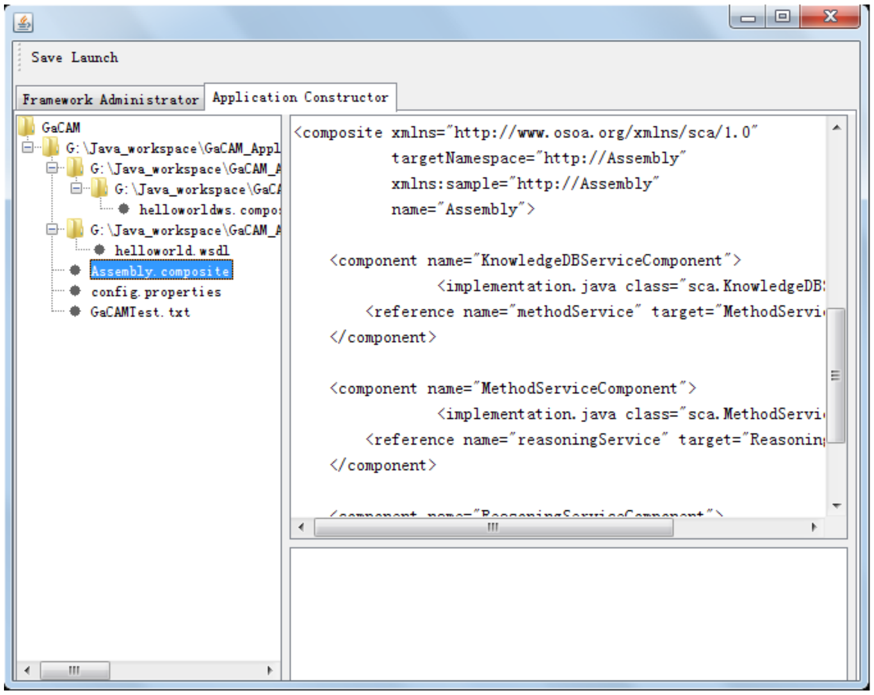


AgentID	Address	Status	Service
JenaAgent@ica.st...	211.144.121.124...	active	JenaFunctions
JessAgent@ica.st...	211.144.121.124...		JessFunctions

Figure 6. The GUI of Framework Administrator

4.4. Application Constructor

The application constructor is in charge of application compositions and reconfigurations. Applications can be constructed and re-constructed according to the change of context. Service Component Architecture (SCA) is already provided by our proposed framework. The GUI of application constructor is shown in Figure 7 and the flow chart of the application composition is shown in Figure 8.



```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
        targetNamespace="http://Assembly"
        xmlns:sample="http://Assembly"
        name="Assembly">

    <component name="KnowledgeDBServiceComponent">
            <implementation.java class="sca.KnowledgeDB
        <reference name="methodService" target="MethodServi
    </component>

    <component name="MethodServiceComponent">
            <implementation.java class="sca.MethodServi
        <reference name="reasoningService" target="Reasonin
    </component>
```

Figure 7. Application Composition

In Figure 7, an XML style service composition configuration file is shown. Through this, applications can be easily composed of existing or newly discovered services. The dynamic reconfiguration is realized through revision of configuration files. A workflow based application composition subsystem is also included in GaCam. Services can also be composed in a sequential manner, as depicted in Figure 8.

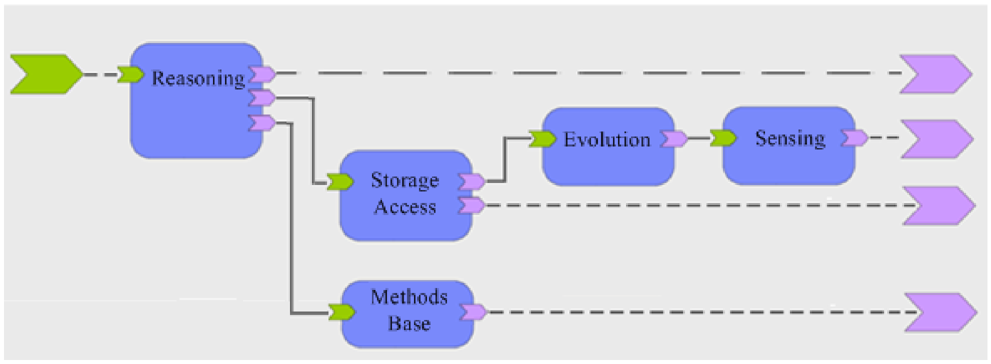


Figure 8. The Flow Chart of the Application Composition

The application constructor achieves the goal of dynamic configuration, and makes services plug and play. This makes substitutions of underlay services won't interfere upper applications. The exploration of bringing in application compositions discloses a way to make frameworks more adaptive and dynamic configurable.

5. Conclusion

In this paper, Open Web Service is proposed to remedy the inherent defects of Web Service technology. After this, Open Web Service is combined with the multi-agent technology, to develop a framework which is more scalable and has better support to dynamic reconfiguration. A prototype system is implemented and illustrated. Several useful tools are also offered either to administrate the framework or to facilitate application developments. Through the development of OWS-MA, an adaptive, reflective, scalable, dynamic reconfigurable, and context aware framework is implementation. Our exploration also proves that the combining of multi-agent and Web Service technology is a promising way to the future development of context-aware applications.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. 60903169 and the Shanghai Science and Technology Development Funds under Grant No. 075107006 and 09510703000, and a Shandong Province Young and Middle-Aged Scientists Research Awards Fund with No.BS2010DX012.

Reference

[1] J. Z. Gu, Communications in Computer and Information Science, vol. 260, **(2011)**, pp. 366.
[2] S. Costantini, L. Mostarda, A. Tocchio and P. Tsintza, IEEE Intelligent Systems, vol. 2, **(2008)**, pp. 34.
[3] Q. Wu, H. Cheng, G. Pan, M. Zhao and J. Sun, IEEE Transactions on Intelligent Transportation Systems, vol. 1, **(2007)**, pp. 121 .

[4] Y. Suo, N. Miyata and H. Morikawa, IEEE Transactions on Knowledge and Data Engineering, vol. 6, **(2009)**, pp. 814.
[5] W3C, "Web Services Architecture," W3C Working Group. Note 11, http://www.w3.org/TR/ws-arch/, **(2004)**.
[6] G. Coulson and P. Grace, WETICE 04. Proceedings of the 13th International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises; 2004 June 14-16; Modena, Italy; **(2004)** pp. 291–296.
[7] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems", PhD thesis. University of Maryland, **(2004)**.
[8] G. S. Blair and G. Andersen, IEEE Distributed Systems Online, vol. 6, **(2001)**, pp. 1.
[9] J. Z. Gu, Journal of East China Normal University (natural science), vol. 5, **(2009)**, pp. 1.
[10] J. Z. Gu, L. He and J. Yang, Communications in Computer and Information Science, vol. 61, **(2009)**, pp. 250.