Service-oriented coordinated intelligent rational agent model for distributed information systems

Wang, Y. D.; Ghenniwa, H.; Shen, W.

Publisher's version / Version de l'éditeur:

The 2009 13th IEEE International Conference on Computer Supported Cooperative Work in Design [Proceedings], pp. 1-6, 2009-04-22

Service-oriented coordinated intelligent rational agent model for distributed information systems

NRCC-51140

Wang, Y.D.; Ghenniwa, H.; Shen, W.

April 2009

A version of this document is published in / Une version de ce document se trouve dans:
The 2009 13th IEEE International Conference on Computer Supported Cooperative Work in Design, Santiago, Chile, April 22-24, 2009, pp.1-6

Service-Oriented Coordinated Intelligent Rational Agent Model for Distributed Information Systems

Ying Daisy Wang[1], Hamada Ghenniwa[2], Weiming Shen[3], Yunjiao Xue[2]
[1]*School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu, China*
ying.daisy.wang@gmail.com
[2]*Dept. of Electrical and Computer Engineering, University of Western Ontario, Canada*
{hghenniwa,yxue24}@eng.uwo.ca
[3]*Institute of Research in Construction, National Research Council Canada*
weiming.shen@nrc.gc.ca

Abstract

This paper presents a Service-Oriented Coordinated Intelligent Rational Agent (SO-CIR-Agent) model to address three design issues of open Cooperative Distributed Systems (CDS): autonomy, distribution, and heterogeneity. This work incorporates the service-oriented design paradigm, agent-oriented design paradigm, and Web service technology as supporting pillars by extending the CIR-Agent model so that agents can survive not only in agent-oriented environments but also in service-oriented environments. The implementation issues of the proposed agent model are discussed at the end.

Keywords: Agent, Service-Oriented, Distributed Systems.

1. Introduction

An open CDS can be constructed from distributed entities that have limited knowledge and are able to perform some functions independently and exercise some degree of authority in sharing their capabilities. These entities are able to work together to achieve individual and/or global goals in some domain. The characteristics and the major design issues of open CDS are autonomy in terms of control, distribution / transparency in terms of information location and working environment, and heterogeneity in terms of information content and implementation technology.

Although researches in distributed systems and artificial intelligence attempted to solve these design issues, the solutions such as Service-Oriented Architecture (SOA) are limited in autonomy while cooperative multi-agent systems (MAS) are limited by the assumption that entities in the environment are wrapped as agents and run on agent platforms.

As a software design paradigm, SOA helps to organize and utilize the distributed capabilities in an open CDS and simplifies the communication between entities by adopting well-defined and highly interoperable communication mechanisms. Therefore, services can be found on heterogeneous distributed platforms while still being accessible across the networks. Thus, SOA solves both the distributed and heterogeneous design issues.

On the other hand, Agent-Oriented (AO) architecture [3] empowers agents with the property of autonomy which means that agents are able to perform tasks without direct intervention from other agents including human agents and they have control of their own internal states and actions. This property addresses the open CDS design issue of autonomy.

At present, although SOA claims to include autonomy as one of its properties, autonomy is still its desirable feature. On the other hand, MAS claims architecture to have a distribution property and the AO architecture to have the capability to solve heterogeneous issues. However, this distribution property is based on the assumption that these distributed entities are all implemented as agents and running on the same type (e.g. FIPA-based) of agent platforms. This assumption is not always true in reality and limits the heterogeneous nature of open CDS. Based on the above insight, the SO-CIR-Agent has been proposed as a service-oriented agent model [9] by extending the CIR-Agent model [1], so that agents can survive not only in agent-oriented environments, but also in service-oriented environments and all three design issues are solved. The integration of SOA and AO technology opens up agent technology towards a service-oriented environment. Therefore, agent technology is proposed to a larger set of possible users and the interesting and advanced work carried out by the agent community can be fruitfully exploited in the area of service-oriented computing, e.g. coordination and collaboration.

The rest of this paper is organized as follows: Section 2 provides a brief literature review; Section 3 introduces the SO-CIR-Agent; Section 4 discusses implementation issues of SO-CIR-Agent; and Section 5 concludes the paper with some perspectives.

2. Literature review

As discussed above, AO architecture is strong in solving the autonomy issue. We take an in-depth review and compare different AO models. An approach in AO systems design views the system as a rational agent having certain *mental attitudes* of Belief, Desire and Intention (BDI), representing the information, motivational, and deliberative states of the agent [7] respectively. The CIR-Agent model previously developed within our group [1] is a design paradigm for cooperative distributed systems. This work extends the CIR-Agent model with SO capability.

A multi-agent system (MAS) is designed not only for achieving autonomy but also for overcoming the distribution and heterogeneity caused by the open nature of CDS [5,8]. The way MAS deals with distribution is based on adoption of Agent Communication Languages (ACL) such as FIPA ACL, and/or agent platforms such as JADE. It effectively resolves these two major design issues while suggests two major implications: (1) agents communicate in the same language, and (2) agents must reside on agent platforms. In other words, agents are not equipped with the capability to communicate with heterogeneous counterparts and they are not survivable in heterogeneous environments. Thus, distribution and heterogeneity are solved to a certain degree by MAS while new concerns are arising right from there.

As mentioned above, SOA helps to organize and utilize the distributed capabilities in an open CDS as well as simplifies the communication by adopting a well-defined and highly interoperable communication mechanism so that services can be discovered on heterogeneous distributed platforms. Web service is an implementation of SOA and is designed to support interoperable machine-to-machine interaction over a network. Its major advantages are the adoption of: (1) XML-based highly interoperable communication / interaction languages, and (2) widely accessible Web technology as the communication backbone which is based on HTTP – a text-based protocol.

Researchers have proposed different approaches to benefit from both sides of the SOA and AO architecture. Some of them investigated the Web services-oriented agent [4] at the implementation level while some others focus on extending the agent platform with Web service capability [2] or the integration between a certain agent platform and a Web service platform [6]. This paper discusses the combination of SOA and AO architecture at the conceptual level and presents a concrete implementation based on CIR-Agent model extension that uses JADE as the agent platform and Web services as SOA implementation. Thus, it gives a high level and broader view for future SOA and AO architecture hybrid software engineering. The implementation is not limited by a certain choice of platforms. Also the agent's survivability in the SO environment no longer depends on the agent platform on which it resides.

3. SO-CIR-Agent

SOA has a higher encapsulation and abstraction power than heterogeneous computational artifacts due to its advantages of autonomy, loose coupling, modularity, and interoperability. Thus it is an appropriate paradigm for open computing environments. However, the service itself is not automatically autonomous. It is rather a desirable characteristic of SOA's conceptual model than the nature of it. On the other hand, AO architecture is promising in realizing complex interactions and coordination behaviors due to its autonomy characteristics. Naturally, we can get a Service-Oriented Agent Model (SOAM) by hybridizing the SOA and AO architecture and benefit from both of them as shown in Figure 1. In a SOAM, AO design provides a focus and cohesive set of service capabilities. The fundamental elements of the environment are services. Software agents capture and implement services as their functionalities and services consider software agents as their owners. A SOAM can be implemented in Web service in terms of service-orientation and CIR-Agent in terms of agent orientation.

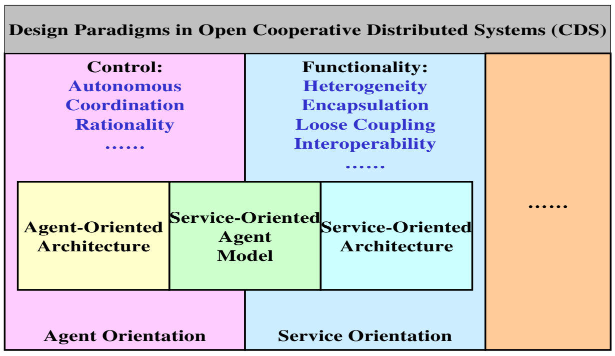

Figure 1. SOAM in Open CDS

The structure of a CIR-Agent is based on the mental state as to achieving a goal. The CIR-Agent architecture is composed of four major components: *knowledge*, *problem-solver*, *interaction* and *communication.* The knowledge component has the mental state of the goals, local world history, coordination knowledge, models of the other agents and communication knowledge. The problem solver is responsible for working out solutions to achieve a goal. The interaction component associates the problems with the corresponding type of interdependencies. The communication component executes the plan so that the agent's acts affect the outside world. In the CIR-Agent model, no global control is allowed, but agents use different interaction devices to resolve the interdependency problem.

To make the CIR-Agent survivable in a SO environment, a particular interconnection of agent components is required to reflect the agent's mental state pattern related to the reasoning about goal achievement. This requires the original CIR-Agent model to be extended in its knowledge and communication components.

3.1. Knowledge Component

The knowledge part is the agent's mental state, which is the information an agent has in its memory. This information includes its business domain knowledge, coordination knowledge, environment knowledge, and communication knowledge as depicted in Figure 2. Two components need to be enriched, namely, external environment knowledge and communication knowledge for service-oriented extension. The external environment knowledge needs to include: (1) the knowledge about other service models, namely, their capability, location, parameters needed to perform a service, and (2) the service platform model that is the operation procedure described in the platform specifications. The communication knowledge model needs to include the knowledge about service communication including service communication languages and protocols.

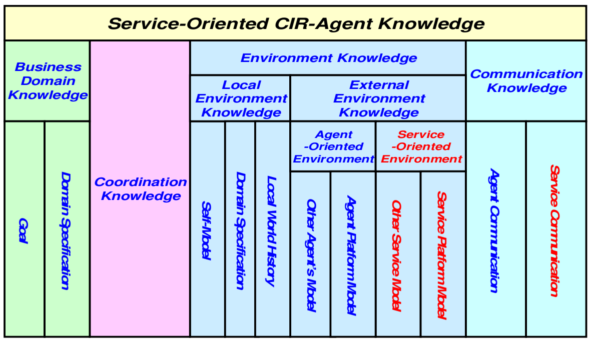

Figure 2. SO-CIR-Agent Knowledge Model

The external environment knowledge of an agent includes the information that an agent has about the external environment, including the knowledge about its AO environment and the knowledge about SO environment.

Model of AO environment, denoted by $MAE^{Ag_i} = \{MA^{Ag_i}, MAP^{Ag_i} | 1 \le i \le m\}$, where MA^{Ag_i} are the models of the other agents in the AO environment and MAP^{Ag_i} is the model of the agent platform Ag_i is running on, and m denotes the number of agents in the environment.

Models of the other agents are denoted by $MA^{Ag_i} = \{M^{Ag_i}_{Ag_l} | 1 \le i \le m, 1 \le l \le m, i \ne l\}$, where $M^{Ag_i}_{Ag_l} \underline{\underline{def}} \langle X_1^{Ag_i}, X_2^{Ag_i}, \dots, X_x^{Ag_i} \rangle$ - the definition of X_s defines the parameters that agent Ag_i might know about agent Ag_l, and m denotes the number of agents in the environment. These parameters might include the capability of managing information and the mental state of an agent in achieving its goals. Such knowledge is usually dynamic and kept in its memory by agents.

An agent usually runs only on one agent platform. The model of the agent platform is denoted by MAP^{Ag_i}. This model is usually reflected in the user's guide or system specification of the agent platform. Information might include the execution procedure that an agent needs to know for registration, and the interface an agent needs to follow to interact with the platform, etc.

Model of SO environment, denoted by $MSE^{Ag_i} = \{MS^{Ag_i}, MSP^{Ag_i} | 1 \le i \le m\}$, where MS^{Ag_i} are the models of the other services in the SO environment and MSP^{Ag_i} is the model of the service platforms Ag_i is involved with, and m denotes the number of agents in the environment.

Models of the other services are denoted by $MS^{Ag_i} = \{M^{Ag_i}_{Sv_l} | 1 \le i \le m, 1 \le l \le n\}$, where $M^{Ag_i}_{Sv_l} \underline{\underline{def}} \langle Y_1^{Ag_i}, Y_2^{Ag_i}, \dots, Y_y^{Ag_i} \rangle$ - the definition of Y_s defines the parameters that agent Ag_i might know about service Sv_l, and m denotes the number of agents in the environment while n denotes the number of services in the environment. These parameters that agent Ag_i might know about service Sv_l include the capability and location of a service, as well as parameters needed to perform the service.

An agent might be involved in more than one service platform. Model of the service platforms is denoted by $MSP^{Ag_i} = \{MSP^{Ag_i}_{Sp_l} | 1 \le i \le m, 1 \le l \le r\}$, where $MSP^{Ag_i}_{Sp_l}$ is the model of a service platform that agent Ag_i is involved with, and m denotes the number of agents in the environment while r denotes the number of service platforms that agent Ag_i is involved with in the environment.

Furthermore, the communication knowledge of an agent includes: Models of agent-oriented communication, denoted by $CA = \{\langle AL_i, O_i \rangle | 1 \le i \le p\}$, where p denotes the number of AO communication languages existing in the environment, AL_i denotes one of the AO communication languages while O_i denotes the ontology pairing with AL_i. Models of service-oriented communication, denoted by $CS = \{\{\langle SL_i, O_i \rangle\}, \{P_j\} | 1 \le i \le q, 1 \le j \le r\}$, where q denotes the number of SO languages existing in the environment, r denotes the number of communication protocols existing in the environment, SL_i denotes the SO communication language existing in the environment while Q_i denotes the ontology pairing with SL_i. Thus, there are two subsets of elements in the set of SO communication model – the SO communication language pair and the communication protocol.

3.2. Communication Component

Two components (Agent-NonAgent and Agent-Environment communication) of the original model need to be extended as depicted in Figure 3. This work adds service as new types of NonAgent entity. Agents need to follow the service communication languages and protocols. This work also adds the SO environment (or service platform) as a new type of environment. It is related to a series of execution procedures whose semantics is based on service platform specifications.

We need to address language, ontology and protocol when talking about communication. The following discussion is based on Web service communication.

Agent-Service Communication. Communication Language – for agents running on a Web Service Platform, they need to communicate with services in SO communication languages, e.g. SOAP for accomplishing a service, WSDL for describing the agent's problem solving capabilities, and UDDI for describing the agents' own locations. Communication Ontology – for agents and services to understand each other correctly, they need to refer to an associated ontology, e.g. OWL, for encapsulation and embodiment of the domain business concepts and rules. Communication Protocol – communication networking protocols, e.g. HTTP, and TCP/IP.

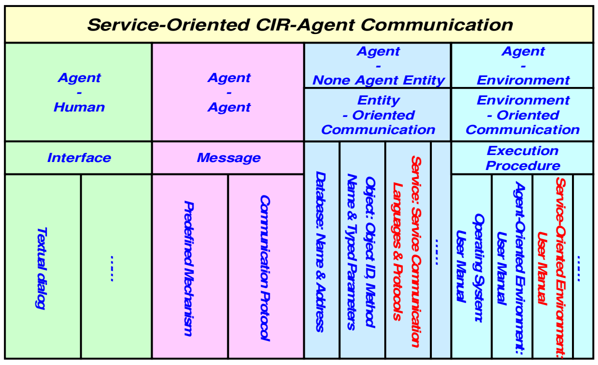
Service-Oriented CIR-Agent Communication
Agent - Human
Agent - Agent
Agent - None Agent Entity
Agent - Environment
Entity - Oriented Communication
Environment - Oriented Communication
Interface
Message
Execution Procedure
Textual dialog
.....
Predefined Mechanism
Communication Protocol
Database: Name & Address
Object: Object ID, Method Name & Typed Parameters
Service: Service Communication Languages & Protocols
.....
Operating System User Manual
Agent-Oriented Environment: User Manual
Service-Oriented Environment: User Manual
.....

Figure 3. SO-CIR-Agent Communication Model

Agent-ServicePlatform Communication. Communication Language – for agents to communicate with SO platforms in operations such as service registration and service look up, in a way (or call it a "language" literally) that the SO platform understands, agents need to follow platform specifications. Communication Ontology – usually the semantics of the communication between any type of service providers/requesters and the SO platform is implicitly reflected in the documentations mentioned in "Communication Language" above. Communication Protocol – usually the SO platform would provide an API for entities to connect with it, and this would serve as the communication protocol between the agents and the SO platforms.

The CIR-Agent communication model is divided into four layers – Goal Layer, Conversion Layer, Message Layer and Physical Layer – to cope with the agent's interaction with other entities. The Goal Layer provides mapping between agent internal/external goal(s) and the conversion layer. The Conversion Layer is a structurally well-formed language used to ensure sending/receiving the intended messages without misunderstanding. The Message Layer provides mapping between the conversion layer and the physical layer, which involves outgoing message construction in ACL and incoming message parsing. The Physical Layer provides a uniform interface to the underlying telecommunication physical layer.

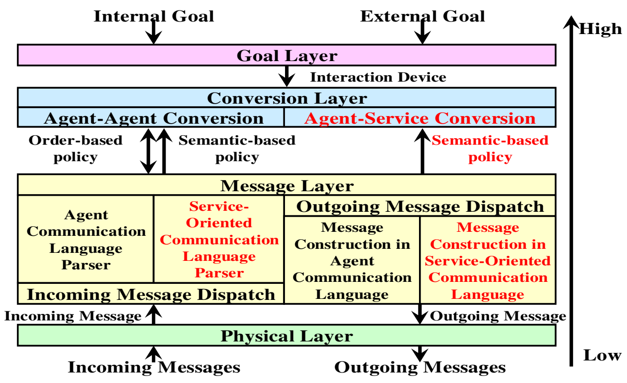

Figure 4. SO-CIR-Agent Communication Layer Abstract Model

Assuming that agents and services are working in the same type of network, it is obvious that we need to have the conversion layer and the message layer extended for SO. The other two layers are not affected as depicted in Figure 4.

Conversion Layer SO extension: The order-based predefined conversion is not applicable in agent-service communication because the commitment to such an order cannot be ensured when agents communicate with heterogeneous types of counterparts. The semantic-based emergent conversion policy allows the agent to use a dynamic order of messages and is based on the interpretation of the received messages. Therefore, the communication conversion mechanism solely relies on the agent's interpretation capability rather than the counterparts' commitment. As long as the agent is equipped with such a message exchanging conversion mechanism, the effectiveness and correctness are ensured.

Message Layer SO extension: When an agent needs to send out a message, the outgoing message is dispatched according to the type of intended receiver first. For a message to be sent to a service, it would be passed to the message construction unit according to SO communication languages, such as SOAP, WSDL and UDDI for Web Services, and then passed to the physical layer. When an agent receives a message from the physical layer, it distinguishes whether the sender is an agent or a service first. For a service type of sender, the newly extended SO Communication Language-based parser would be applied.

4. Implementation

When an instance of a SO-CIR-Agent is deployed, among the four components of the agent model, the *Knowledge* component plays an essential role. The remaining three components (Problem Solver, Interaction Device and Communication) are deployed based on an agent's knowledge about itself and the environment. Based on this insight, this work proposes a Knowledge-Driven Self-Deployment Algorithm that allows an instance of a SO-CIR-Agent being deployed driven by its knowledge.

For a SO-CIR-Agent Ag_i, it is represented as: $Ag_i = \{K^{Ag_i}, PS^{Ag_i}, ID^{Ag_i}, C^{Ag_i}\}$, where K^{Ag_i} is the *Knowledge* component, denoted by $K^{Ag_i} = \left\{\begin{matrix} G^{Ag_i}, D^{Ag_i}, CK^{Ag_i}, SK^{Ag_i}, \\ LH^{Ag_i}, MA^{Ag_i}, MAP^{Ag_i}, MS^{Ag_i}, \\ MSP^{Ag_i}, CA^{Ag_i}, CS^{Ag_i} \end{matrix}\right\}$ where G^{Ag_i} is the *Goal* knowledge, D^{Ag_i} is the *Domain Specification* knowledge, CK^{Ag_i} is the *Coordination Knowledge*, denoted by $CK^{Ag_i} = \left\{\begin{matrix} RSK^{Ag_i}, CRK^{Ag_i}, SCK^{Ag_i}, \\ RAK^{Ag_i}, AK^{Ag_i}, KUK^{Ag_i} \end{matrix}\right\}$, where RSK^{Ag_i} is the *Resource Scheduling* knowledge, CRK^{Ag_i} is the *Conflict Resolution* knowledge, SCK^{Ag_i} is the *Synchronization* knowledge, RAK^{Ag_i} is the *RedundancyAvoidance* knowledge, AK^{Ag_i} is the *Assignment* knowledge, KUK^{Ag_i} is the *Knowledge Update* knowledge, SK^{Ag_i} is the *Self Model* knowledge, it points to SO-CIR-Agent instance Ag_i, LH^{Ag_i} is the *Local History* knowledge, MA^{Ag_i} is the *Model of other Agent* knowledge, MAP^{Ag_i} is the *Model of Agent Platform* knowledge, MS^{Ag_i} is the *Model of other Service* knowledge, MSP^{Ag_i} is the *Model of Service Platform* knowledge, CA^{Ag_i} is the *Agent Communication* knowledge, CS^{Ag_i} is the *Service Communication* knowledge, PS^{Ag_i} is the *Problem Solver* component which takes self model SK^{Ag_i} for initiation, ID^{Ag_i} is the *Interaction Device* component, denoted by $ID^{Ag_i} = \left\{\begin{matrix} RS^{Ag_i}, CR^{Ag_i}, S^{Ag_i}, \\ RA^{Ag_i}, A^{Ag_i}, KU^{Ag_i} \end{matrix}\right\}$, where RS^{Ag_i} is the *Resource Scheduling* device which takes SK^{Ag_i} for initiation, CR^{Ag_i} is the *Conflict Resolution* device which takes SK^{Ag_i} for initiation, S^{Ag_i} is the *Synchronization* device which takes SK^{Ag_i} for initiation, RA^{Ag_i} is the *RedundancyAvoidance* device which takes SK^{Ag_i} for initiation, A^{Ag_i} is the *Assignment* device which takes SK^{Ag_i} for initiation, KU^{Ag_i} is the *Knowledge Update* device which takes SK^{Ag_i} for initiation, C^{Ag_i} is the *Communication* component, denoted by $C^{Ag_i} = \{AHC^{Ag_i}, AAC^{Ag_i}, ASC^{Ag_i}, AEC^{Ag_i}\}$, where AHC^{Ag_i} is the *Agent-Human Communication* device which takes SK^{Ag_i} for initiation, AAC^{Ag_i} is the *Agent-Agent Communication* device, ASC^{Ag_i} is the *Agent-Service Communicaton* device, AEC^{Ag_i} is the *Agent-Environment Communication* device, denoted by $AEC^{Ag_i} = \{OSC^{Ag_i}, APC^{Ag_i}, SPC^{Ag_i}\}$, where OSC^{Ag_i} is the *Agent-OS Communication* device which is the only default component of Ag_i, and it has a read() method, read() takes knowledge name as input and returns with the corresponding knowledge, APC^{Ag_i} is the *Agent-AgentPlatform Communication* device, SPC^{Ag_i} is the *Agent-ServicePlatform Communication* device. Some Knowledge component member is necessary for agent deployment while others are not. Instead, they are essential for an agent to be functional during its life cycle. Assuming the agent's deployment capability is guaranteed when necessary knowledge is available, the logical relationship between the deployment necessary member of a Knowledge component and the member of the rest of a SO-CIR-Agent components according to their semantics and intended use are represented as follows:

$(OSC^{Ag_i} \wedge CA^{Ag_i} \wedge MAP^{Ag_i}) \rightarrow (AAC^{Ag_i} \wedge APC^{Ag_i}$	(1)
$(OSC^{Ag_i} \wedge CS^{Ag_i}) \rightarrow ASC^{Ag_i}$	(2)
$(OSC^{Ag_i} \wedge MSP^{Ag_i}) \rightarrow SPC^{Ag_i}$	(3)
$(OSC^{Ag_i} \wedge G^{Ag_i} \wedge D^{Ag_i}) \rightarrow PS^{Ag_i}$	(4)
$(OSC^{Ag_i} \wedge RSK^{Ag_i}) \rightarrow RS^{Ag_i}$	(5)
$(OSC^{Ag_i} \wedge CRK^{Ag_i}) \rightarrow CR^{Ag_i}$	(6)
$(OSC^{Ag_i} \wedge SCK^{Ag_i}) \rightarrow S^{Ag_i}$	(7)
$(OSC^{Ag_i} \wedge RAK^{Ag_i}) \rightarrow RA^{Ag_i}$	(8)
$(OSC^{Ag_i} \wedge AK^{Ag_i}) \rightarrow A^{Ag_i}$	(9)
$(OSC^{Ag_i} \wedge KUK^{Ag_i}) \rightarrow KU^{Ag_i}$	(10)
$\left(\begin{matrix} (\exists x(x \in \{RSK^{Ag_i}, CRK^{Ag_i}, SCK^{Ag_i}, RAK^{Ag_i}, AK^{Ag_i}, KUK^{Ag_i}\})) \\ \wedge G^{Ag_i} \wedge D^{Ag_i} \\ \wedge CA^{Ag_i} \wedge MAP^{Ag_i} \wedge CS^{Ag_i} \wedge MSP^{Ag_i} \end{matrix}\right) \rightarrow K^{Ag_i}$	(11)
$\exists x(x \in \{RS^{Ag_i}, CR^{Ag_i}, S^{Ag_i}, RA^{Ag_i}, A^{Ag_i}, KU^{Ag_i}\}) \rightarrow ID^{Ag_i}$	(12)
$\{AHC^{Ag_i}, AAC^{Ag_i}, ASC^{Ag_i}, OSC^{Ag_i}, APC^{Ag_i}, SPC^{Ag_i}\} \rightarrow C^{Ag_i}$	(13)
$(K^{Ag_i} \wedge PS^{Ag_i} \wedge ID^{Ag_i} \wedge C^{Ag_i}) \rightarrow Ag_i$	(14)

The objective of this algorithm is to take K^{Ag_i} as input, and generate the rest of a SO-CIR-Agent components (PS^{Ag_i}, ID^{Ag_i} and C^{Ag_i}). The expected output should be a component collection in the normalized form of $Ag_i = \{K^{Ag_i}, PS^{Ag_i}, ID^{Ag_i}, C^{Ag_i}\}$, like:

$$Ag_i = \left\{\begin{matrix} G^{Ag_i}, D^{Ag_i}, \\ (\exists x(x \in \{RSK^{Ag_i}, CRK^{Ag_i}, SCK^{Ag_i}, RAK^{Ag_i}, AK^{Ag_i}, KUK^{Ag_i}\})), \\ SK^{Ag_i}, LH^{Ag_i}, \\ MA^{Ag_i}, MAP^{Ag_i}, MS^{Ag_i}, MSP^{Ag_i}, \\ CA^{Ag_i}, CS^{Ag_i}, \\ PS^{Ag_i}, \\ (\exists x(x \in \{RS^{Ag_i}, CR^{Ag_i}, S^{Ag_i}, RA^{Ag_i}, A^{Ag_i}, KU^{Ag_i}\})), \\ AHC^{Ag_i}, AAC^{Ag_i}, ASC^{Ag_i}, APC^{Ag_i}, SPC^{Ag_i} \end{matrix}\right\}$$

5. Conclusions

This paper presents SO-CIR-Agent as a Service-Oriented agent model in solving the three design issues of Open CDS, namely, autonomy, distribution and heterogeneity. This work involves a number of emerging technologies including Web services and intelligent agents. The research starts from a conceptual level investigation of service-oriented agent model as an infrastructure to carry out services in open CDS. The advantages of such an agent model include: (1) AO design empowers the model with autonomy; (2) SOA endows the model with heterogeneity, encapsulation, loose coupling and interoperability; (3) most importantly, agents are robust so that individual agents can survive in both AO and SO environments.

Considering the concrete implementation based Web service and CIR-Agent model, this work proposes a Knowledge-Driven Self Deployment algorithm to guide the agent deployment progress. The proposed SO-CIR-Agent is generic and can be applied to many open CDS application domains, such as intelligent personal assistant services, e-business, healthcare, and resource management. In future work, we intend to validate the proposed model through comprehensive scenarios in different application domains and on different agent platforms.

6. References

[1] Ghenniwa, H., Kamel, M., 2000. Interaction Devices for Coordinating Cooperative Distributed Systems, *Automation and Soft Computing*, 6(2), 173-184.

[2] Greenwood, D., Calisti, M., 2004. An Automatic, Bi-Directional Service Integration Gateway, *IEEE Systems, Cybernetics and Man Conference*, 10-13 October, The Hague, Netherlands.

[3] Jennings, N., Wooldridge, M., 2001. Agent-Oriented Software Engineering, *Handbook of Agent Technology*, J. Bradshaw (Eds.), AAAI/MIT Press.

[4] Li, Y., Ghenniwa, H., Shen, W., 2004. Agent-Based Web Services Framework and Development Environment, *Computational Intelligence*, 20(4), 678-692.

[5] Marik, V., Muller, J., Pechoucek, M., Eds., 2003. *Multi-Agent Systems and Applications,* Springer-Verlag Heidelberg, pp. 626-635.

[6] Nguyen, X.T., Kowalczyk, R., 2007. WS2JADE: Integrating Web Service with Jade Agents, *Service-Oriented computing: Agents, Semantics, and Engineering*, Springer Berlin/Heidelberg, pp. 147-159.

[7] Rao, A.S., Georgeff, M.P., 1995. BDI agents: From theory to practice, *Proceedings of ICMAS-95*, pp. 312-319, Menlo Park, California, June, AAAI Press.

[8] Wooldridge, M., 2002. An Introduction to Multi-Agent System, John Wiley & Sons.

[9] Wang, Y.D., Ghenniwa, H., Shen, W., 2008. An Amphibian Service-Oriented Agent Model for Cooperative Distributed Systems, IEEE International Conference on e-Business Engineering, Xian, China, 22-24 October.