Mobile Business Agents Model and Architecture

Haeng Kon Kim

Department of Computer Engineering, Catholic University of Daegu,
Kyungsan, Kyungbuk, 712-702, Seoul of Korea
hangkon@cu.ac.kr

Abstract. Agent-component technology and agent-oriented software engineering have the potential to be more powerful than traditional. Most agent and e-service systems offer several capacities that work together to provide unprecedented flexibility and promise to be more effective at handling the resulting software's evolution and distribution. Therefore, in order to support agent service or agent based business application and system there is the necessity of research about agent development based component. In this paper, we identify and classify the general and e-business oriented agent affecting CBD. We suggest the e-business agent oriented component reference architecture. We also propose systemical development process using AUML(Agent Unified Modeling Language) and design pattern technology to analysis, design and develop e-business agent. Finally we describe how these concepts may assist in increasing the efficiency and reusability in business application and e-business agent development.

Keywords: E-Business Agent, Agent Classification, Component Architecture, CBD, Agent Design Patten.

1 Introduction

Agent-oriented techniques represent an exciting new means of analyzing, designing and building complex software systems. They have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasibly be tackled. As the demand for more flexible, extensible, and robust Web-based enterprise application systems accelerates, adopting new software engineering methodologies and development strategies becomes critical. These strategies must support the construction of enterprise software systems that assemble highly flexible software components written at different times by various developers[1, 2].

In this paper, we identify the primary and general attribute from existing application and classify the agents form e-business domain as a sub research to develop e-business agent based component. Through all over this, common area is extracted both general agent and e-business agent and e-business agent oriented component with reference architecture. We also propose systemical development process using AUML and design pattern technology to analysis, design, and develop e-business agent. Component reference architecture through agent domain classification is based on component development life cycle. Moreover, the development of e-business agent and system can be obtained the efficiency through component technology.

T.-h. Kim et al. (Eds.): FGIT 2010, LNCS 6485, pp. 676–685, 2010.

2 Related Works

2.1 Basic Characteristics of E-Business Agents

An agent must have a model of its own domain of expertise and a model of the other agents that can provide relevant information. The awareness model of an information agent does not need to contain a complete description of the other agents' capabilities, but rather only those portions that may be directly relevant when handling a request that cannot be serviced locally. In general, we require that intelligent business agents possess distinguishing characteristics described in the following paragraphs[3].

- Delegation abilities: The central idea underlying agents is that of delegation. The owner or user of an agent delegates a task to the agent and the agent autonomously performs the task of behalf of the user. Alternatively, a business agent may decompose the task and delegate parts of it to other agents, which perform the subtasks and report back to the business agent. The agent must be able to communicate with the user or other agents to receive its instructions and to provide results of its activities.
- Agent communication languages and protocols: A business agent is an autonomous entity, hence it must negotiate with other agents to gain access to other sources and capabilities. To enable the expressive communication and negotiation required and organize communications between agents, a language that contains brokering performatives can be particularly useful. Some general examples of agent development environments include the agent builder and the agent library.
- Self-representation abilities: One of the most challenging problems is for agents to express naturally and directly business and system aspects and then combine these into a final meaningful application or implementation. This results in self-describing, dynamic, and reconfigurable agents that facilitate the composition of large-scale distributed applications, by drawing upon business processes and the functionality of existing information sources. Such ideas can benefit tremendously from techniques found in reflection and meta-object protocols.

2.2 AUML(Agent Unified Modeling Language)

The current UML is sometimes insufficient for modeling agents and agent-based systems. However, no formalism yet exists to sufficiently specify agent-based system development. To employ agent-based programming, a specification technique must support the whole software engineering process[4]. Both FIPA(Foundation for Intelligent Physical Agents) and the OMG Agent Work Group are exploring and recommending extensions to UML[1]. The AUML present a subset of an agent based extension to the standard UML for the specification of AIP(Agent Interaction Protocols) and other commonly used agent based notions. An AIP describes a communication pattern as an allowed sequence of messages between agents and the constraints on the content of those messages.

Interaction protocols were chose because they are complex enough to illustrate the nontrivial use of AUML and are used commonly enough to make this subset of AUML useful to other researchers. Agent interaction protocols are a good example of software patterns that are ideas found useful in one practical context and probably

useful in others. A specification of an AIP provides an example or analogy that we might use to solve problems in system analysis and design. AUML suggest a specification technique for AIPs with both formal and intuitive semantics and a user-friendly graphical notation. The semantics allows a precise definition that is also usable in the software-engineering process. The graphical notation provides a common language for communicating AIPs[5].

3 E-Business Agent Oriented CBD Reference Architecture

Component reference architecture classifies general agent which analysis existing agent system based on primary property. We can classify the agents by domain attributes, usages and requirement in e-business system. The common area identified from general agent type and e-business agent domain. The reference architecture is constructed based on identified common area and support components for e-business agent development. The architecture offers guideline for adaptable component analysis and design. It also supports component deployment and management.

3.1 Classification

The agents found in systems have special requirements: they must execute as software, hardware, robotics, or a combination of these. Agent developers have identified several forms of agents that are important for application development. The list of agent characteristics presented earlier addresses some of these requirements. Additionally, since agent system has special needs, software and hardware-related forms must be considered. We attempts to palace existing agents into different agent classes. Then, its goal is to construct component reference architecture for e-business agent. We consider both primary attributes and business attributes in existing agent system.

3.1.1 Classification of General Agent

The type of agent, definition, and name, which are used in existing agent system, are circulate in various way. We identify fourteen different types of agents with attribute as in figure 1. We would overview them in terms of some or all of the following.

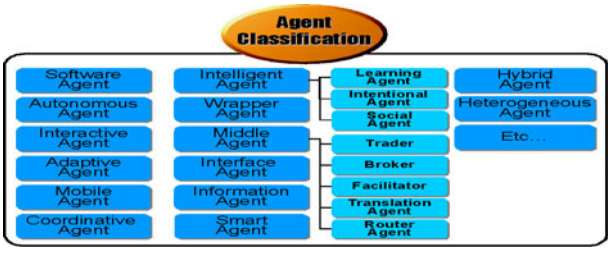

Fig. 1. General Agent Classification

Software agent is defined as *an autonomous software entity that can interact with its environment*. This means that they are autonomous and can react with other entities, including humans, machines, and other software agents in various environments

and across various platforms. When an agent has a certain independence from external control, it is considered autonomous. Without any autonomy, an agent would no longer be a dynamic entity, but rather a passive object such as a part in a bin or a record in a relational table. Interactive agents can communicate with both the environment and other entities and can be expressed in degrees. An agent is considered *adaptive* if it is capable of responding to other agents and/or its environment to some degree. At a minimum, this means that an agent must be able to *react* to a simple stimulus, predetermined response to a particular event or environmental signal.

While stationary agents exist as a single process on one host computer, mobile agents can pick up and move their code to a new host where they can resume executing. The rationale for mobility is the improved performance that can sometimes be achieved by moving the agent closer to the services available on the new host. Human organizations exist primarily to coordinate the actions of many individuals for some purpose. Using human organizations as an analogy, systems involving many agents could benefit from the same pattern. Some of the common coordinative agent applications involve supply chains, scheduling, vehicle planning, problem solving, contract negotiation, and product design.

After decades, the term *intelligent* has still not been defined for artificial system and applying the term now to agents may not be appropriate. Most tend to regard the term *agent* and *intelligent agent* as equivalent. Perhaps this is just an attempt to communicate that agents have more power that conventional approaches. Some kinds of intelligent agents are learning agent, intentional agent and social agent. Client agents can relay commands to the wrapper agent and have them invoked on the underlying services. The role provided by the wrapper agent provides a single generic way for agents to interact with non-agent software systems.

Broadly, agentized middleware including agentized common and basic object services. Some kinds of middle agents are trader, broker, facilitator, translation agent and router agent. An interface agent is a program that is able to operate within a user interface and actively assist the user in operating the interface and manipulating the underlying system. An interface agent is able to intercept the input from the user, examine it, and take appropriate action. Agents in the interface can function as a bridge between domain knowledge about the data management systems and the user.

Smart agents are supposed to be able to learn as they react and/or interact with their external environment, so that, with time, their performance increases. Hybrid agents refer to those whose constitution is a combination of two or more agent *philosophies* within a singular agent. The key *hypothesis* for having hybrid agents or architectures is the belief that, for some application, the benefits accrued from having the combination of philosophies within a singular agent is greater than the gains obtained from the same agent based entirely on a singular philosophy. Heterogeneous agent systems, unlike hybrid systems described in the preceding section refers to an integrated set-up of at least two or more agents, which belong to two or more different agent classes.

3.1.2 Agent Classification of e-Business Agent

The current kinds of applications that employ agents is still limited. Once the concepts become more accepted and more tools become available, the agent-based approach will become more embedded it e-business domain and applications. Figure 2 shows an agent classification with e-business attribute and function.

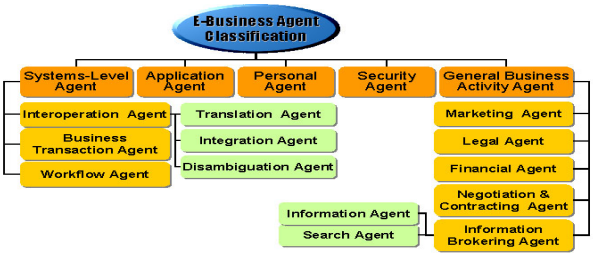

Fig. 2. E-Business Agent Classification

In a e-business environment it is necessary to organize agents into different categories depending on their functionality and competencies. The five basic type of agents can be distinguished as described here. System-level agents exist on top of the distributed objects infrastructure, typically implemented in CORBA by means of the IIOP, which provides objects with transparent access not only to other application objects but also to such facilities as transaction processing, permanent object storage, event services, and the like. Agent solutions are deployed as an extension of the distributed object foundation and may assist in accomplishing the following systems related tasks. Some of the advanced functionality agents required providing support for e-commerce and interoperation of open market business processes are described here. It includes *interoperation agent, business transaction agent, work flow agent*.

A business-to-business e-commerce application is a networked system that comprises a large number of application agents. Each agent is specialized to single area of expertise and provides access to the available information and knowledge sources in that domain and works cooperatively with other agents to solve a complex problem in that vertical domain. This results in the formation of clusters of information sources around domains of expertise handled by their respective agents.

Personal agents work directly with users to help support the presentation, organization, and management of user profile, requests, and information collections. A personal agent gives its user easy and effective access to profile related specialized services and information widely distributed on the Web. The user's agent observes and monitors the actions taken by the user in the interface and suggests better ways to perform the task. These agents can assist users in forming queries, finding the location of data, and explaining the semantics of the data, among other tasks.

The activities and functions of e-business need certain basic agent technology support that is likely to become the basis for developing standard digital agents for e-business. General business agents perform a large number of general commerce support activities that can be customized to address the needs of a particular business organization. It includes *marketing, legal, negotiation, information brokering agent*.

E-business communication need to be guarded by specially designed agents that provide the security services required for the conduct of e-business. Agent support for secure e-business can be segmented into five distinct categories: authentication, authorization, data integrity, confidentiality, and non-repudiation.

- Authentication agents can be used to identify the source of a message sent over the Internet.

- Authorization agents may control access to sensitive information once identity has been verified. Thus, certain transactions may need to be partly accessible to certain parties, while the remainder of the transaction is not. The transaction workflow and authorization agents can coordinate these tasks.
- Secure transactions should guarantee that a message has not been modified while in transit. This is commonly known as integrity and is often accomplished through digitally signed digest codes. Transactions should also guarantee confidentiality.
- Confidentiality refers to the use of encryption for scrambling the information sent over the Internet and stored on servers so that eavesdroppers and interlopers cannot access the data.
- Non-repudiation is of critical importance for carrying out transactions over the Internet. It consists of cryptographic receipts that are created so that the author of a message cannot falsely deny sending a message.

3.2 Component Reference Architecture for E-Business Agent Development

In order to construct component reference architecture, agent is classified in general agent type and e-business function attribute. Figure 3 is a component and meta architecture of based on all above described for e-business agent.

Reference architecture is consisted of dimension, which has 15 general types and 11 concrete business agent types with domain oriented component architecture. These two classification areas tend to be independent for each cross-referenced. Each area has its own horizontal and vertical characteristics. General agent types are corresponding to agent platform and application. It is possible to develop agent system or application by the referencing architecture. The technology of agent can be applied to business domain. Developed component is classified by the reference architecture and is placed according to general agent type and business attribute. In case agent is applied to the agent system or business domain, system is possibly to build up by identifying component related to business domain and combining it.

Fig. 3. CBD Reference Architecture of E-Business Agent

4 Agent Component Development Process Based Architecture

As we suggested CBD reference architecture in previous chapter, component development process based architecture is a set of activities and associated results, which

lead to the production of a component as in figure 4. These may involve the development of component from UML model.

In figure 4, architecture is at the center of analysis, design, component development, this process applies and designs architecture from early domain analysis phase to component implementation. In addition, we consider systemical development process using AUML and design pattern technology to analysis, design, and develop e-business agent. The domain analysis specification, design model, implemented component, which are produce in process, are stored in the repository[6].

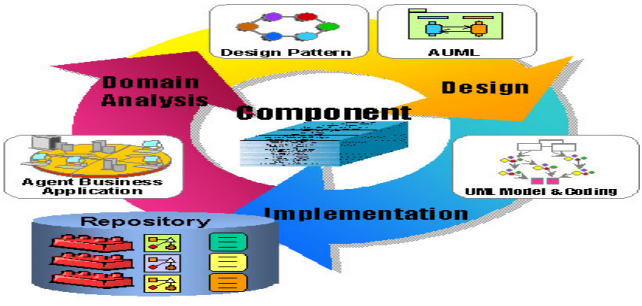

Fig. 4. Component development process

4.1 Agent Domain Analysis Phase

The requirement of agent should be first identified in desired business system. The primary property of agent should be analyzed after that the description for specific agent platform and the sorts of essential properties should be understood. At the same time, it is very important to consider weather the requirement, which is already defined, is corresponding to agent type in reference architecture and what business concept is focused on.

All over those things make high understanding for domain requirement and become referenced to define agent attribute. Selecting of component domain can easily identify design pattern in design phase and easily deploy component. Domain analysis is presented on entire domain concept and scenario using activity diagram. Requirement analysis is defined through use case diagram, and use case description.

4.2 Agent Design Phase

The e-business agent with adaptable component is designed based on domain requirement. Attribute and behavior are defined using class diagram for component, which is expected to be implemented depending on agent type. The definition of component interface is presented on sequence diagram. Contract specification to describe pre-condition, post-condition, and interface properties show the relationship between components. There are two considerations depending on agent property and design technology on design phase. First, part of related to agent interact protocol use AUML notation. And agent interact protocol is described communication pattern. This proposes three levels for the protocols presentation method of agent.

- Overall protocol level: There are two techniques that best express protocol solutions for reuse; package diagram and templates.
- Interactions among agents level: There are presented through UML's dynamic model; sequence, collaboration, activity and state diagram.
- Internal agent processing level: At the lowest level, requires spelling out the detailed processing that takes place within an agent in order to implement the protocol. This layer preset to use activity diagram and state charts.

Second, design pattern can be applied to previously identified area in reference architecture. Figure 5 is design pattern matrix based on meta architecture of component reference architecture and design pattern is identified in matrix. Design pattern is made considering agent functionality and added on other information for component development. Moreover, the concurrency of architecture can be acquired by constructing pattern library applying component reference architecture like development process done. CBD Reference architecture is concern on component, which is supposed to be implemented though analysis and design phase, also possibly apply to entire lifecycle. Figure 6 shows the conceptual process, which are domain analysis, applying design pattern and constructing component-based architecture.

E-Business Agent / Agent Type	System-Level Agent(00)			General Business Activity Agent(10)					Personal Agent (20)	System Level Agent (30)	Security Agent (40)
	Interoperation Agent (01)	Business Transaction Agent (02)	Workflow Agent (03)	Marketing Agent (11)	Legal Agent (12)	Financial Agent (13)	Negotiation /Contracting Agent (14)	Information Brokering Agent (15)			
Software Agent(SWA)											
Autonomous Agent (AUA)											
Interactive Agent (INA)											
Adaptive Agent (ADA)											
Mobile Agent (MBA)											
Coordinative Agent (COA)											
Intelligent Agent (ITA)											
Wrapper Agent (WRA)											
Middle Agent (MOA)											
Interface Agent (IFA)											
Information Agent (IMA)											
Smart Agent (SMA)											
Hybrid Agent (HYA)											
Heterogeneous Agent (HGA)											
etc											

Fig. 5. Design pattern reference matrix

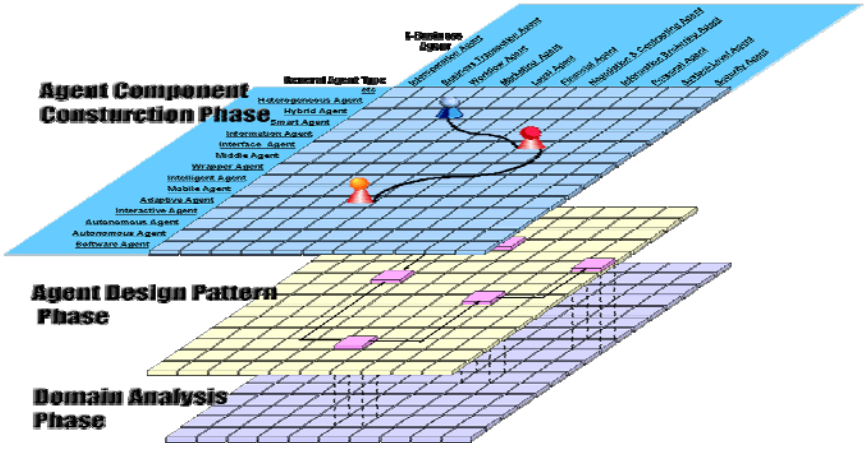

Fig. 6. Agent Development Based Component Reference Architecture

5 Conclusion and Future Works

Agent-oriented technology can help enable the development of e-business agents, which are the next higher level of abstraction in model-based solutions to e-business applications. This technology allows the development of rich and expressive models of an enterprise and lays the foundation for adaptive, reusable business software. Agent-oriented technology can be leveraged to enhance enterprise modeling as well as to offer new techniques for developing applications and infrastructure services.

In this paper, general agent type is classified in 15 categories according to role. e-business agent is classified in 11 categories according to adaptable domain. CBD reference architecture is constructed in 2 dimension based on these categories. In addition, we propose systemical development process based on architecture. This process applies and designs architecture from early domain analysis phase to component development. Design pattern matrix is made in the same architecture mode in component design so that there is a benefit to reduce development time and to have high reusability of design concept. Component reference architecture through agent domain classification is based component development life cycle. Moreover, the development of e-business agent and agent-oriented system can obtain the efficiency through component reuse. In the future work, there needs more study about component integration based CBD reference architecture for e-business agent and agent application system. We also are going to study on the contracting of e-business agent with CBD modeling methodology.

Acknowledgement

“This works supported Korea National Research Foundation (NRF) granted by the Korea Government Scientist of Regional University No.R 2010-0017089”.

References

1. Jennings, N.R., Wooldridge, M.: Agent-Oriented Software Engineering. In: Proceeding of IEA/AIE 1999, pp. 4–10 (2009)
2. Odell, James (eds.): Agent Technology, OMG, green paper produced by the OMG Agent Working Group (2010)
3. Papazoglou, M.P.: Agent-Oriented Technology in support of E-Business. Communications of the ACM 44(4), 71–77 (2010)
4. Odell, J., Van Dyke Parunak, H., Bauer, B.: Extending UML for Agents. In: Proceeding Of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence (2009)
5. Bauer, B., Müller, J.P., Odell, J.: Agent UML: A Formalism for Specifying Multiagent Interaction. In: Proceeding of 2000 Agent-Oriented Software Engineering, pp. 91–103 (2001)
6. Kim, H.K.: Component Repository and Configuration Management System, ETRI Final Research Report (2000)
7. Nwana, H.S.: Software Agents: An Overview, Software Agent Technologies (1996)

8. Kim, H.K., Han, E.J., Shin, H.J., Kim, C.H.: Component Classification for CBD Repository Construction. In: Proceeding of SNPD 2000, pp. 483–493 (2000)
9. Griss, M.L., Por, G.: Accelerating Development with Agent Components. IEEE Computer 34(5), 37–43 (2001)
10. Brereton, P., Budgen, D.: Component-Based Systems:A Classification of Issues. IEEE Computer 33(11) (2000)
11. Aridor, Y., Lange, D.B.: Agent Design Patterns: Elements of Agent Application Design. In: Proceeding of Autonomous Agents 1998, pp. 108–115 (1998)
12. Heineman, G.T., Councill, W.T.: Component-Based Software Engineering. Addison-Wesley, Reading (2001)
13. Jennings, N.R.: On agent-based software engineering. International Journal of Artificial Intelligence 117(2), 277–296 (2003)
14. Park, K., Kim, J., Park, S.: Goal based agent-oriented software modeling. In: Proceeding of the Seventh Asia-Pacific Software Engineering Conference(APSEC 2000), December 2000, pp. 320–324 (2000)
15. Hara, H., Fujita, S., Sugawara, K.: Reusable Software Components based on an Agent Model. In: Proceeding of 7th International Conference on Parallel and Distributed Systems Workshops, July 2000, pp. 447–452 (2000)