Development of Multi Agent for Adapting Convergence Applications

Younky Chung

[*] *Department of Computer Engineering, Kyungil University, Republic of Korea*
ykchung@kiu.ac.kr

Abstract

Mobile agents provide an attractive conceptual framework for mobile device-based computing – small threads migrating from server to server, performing their functions, there are still many difficulties that must be addressed. These difficulties are currently preventing the widespread adoption of the technology. Some of the key problems include: security, user and provider psychological resistance, infrastructure integration, interoperability and reliability.

Mobile computers and devices may operate in a variety of environments with different security schemes. In this paper, we present a mobile agent for convergence applications model. The model supports the flexible and extensible application specific convergence applications measures required by mobile computers and devices in distributed system. We gave a full description of mobile agent and its migration modes, after discussing the possible convergence applications, the model is explained at length and security analysis is also given in the way of collaboration requirements, mobility requirements and execution requirements.

Keyword: *mobile agent, convergence applications, distributed system, dynamic*

1. Introduction

An agent is an atomic autonomous entity that is capable of performing some useful function. The functional capability is captured as the agent's services. A service is the knowledge level analogue of an object's operation. The quality of autonomy means that an agent's actions are not solely dictated by external events or interactions, but also by its own motivation. We capture this motivation in an attribute named purpose. For example, the purpose will influence whether an agent agrees to a request to perform a service and also the way it provides the service [1, 2].Conventional networking technology and the Internet makes information access and wide area communication much easier than ever before. Management of very large, inter-organizational distributed systems cannot be centralized; it must be distributed to reflect the distribution of the system being managed. Especially, the growing infrastructure permits mobile computer user access to information anywhere and at any time. To make networked mobile computers effective in a distributed network, the systems must employ convergence applications provisions to ensure reliability, privacy, authorization, flexibility and extensibility are also requested. There are several deficiencies in traditional security systems such as firewall-based approaches; they are listed as follows: there is a lack of dynamic design, traditional security systems lack dynamic security policies and corresponding enforcement mechanisms, thus they prevents the evolution and adaptation of security policies in rapidly changing environments such as those required for commercial and military collaborative projects. Furthermore, it is not applicable in mobile environment as application of mobile computers/devices is developing rapidly, firewall-based model is challenged by such devices in view of security, which is to say, approaches based on firewall done fit well in a mobile computing environment. There isn't an easy way to maintain the desired defense along the perimeter while still letting mobile computers do useful things.

There exists contradiction between complex security function and limited usable resource. On the other hand, resource limitations of mobile devices make many existing complex security approaches prohibitive to use. Even though such systems can be used, it is still a far-fetched way as the efficiency of systems is too low. Compared to existing approaches, the agent-based one is especially suitable for mobile computers. In virtue of the portability of agent, mobile users can delegate mobile security agents to a trusted mobile host to perform sophisticated resource intensive operations like authentication and encryption. In addition, these agents can also adapt to a changing network and computation environment by downloading localized security policies and mechanisms dynamically from the nearby mobile hosts. On the other hand, mobile security agents can be uploaded into mobile hosts as well to perform application and user specific security function. In that way, the implementation of security systems on a mobile host can be very lightweight and highly optimized. From the security perspective, the agent-based approach also has the advantages of fine granularity, independence and extension.

In this paper, we present a mobile agent-based convergence applications model. The model supports the flexible and extensible application specific convergence applications measures required by mobile computers and devices in distributed system. We gave a full description of mobile agent and its migration modes, after discussing the possible convergence applications, the model is explained at length and security analysis is also given in the way of collaboration requirements, mobility requirements and execution requirements. This paper also describes a dynamic agent-base strategy for improving the security of distributed system. The dynamic agent-based security model is under active development. The following sections describe: [3, 4].

-Attributes and constitution of agent in mobile environment, three paradigms of mobile agent migration are also presented.

-Security problems in mobile agent-based model Description of agent-based security model

-Security analysis of the model in the way of collaboration requirements, mobility requirements and execution requirements.

2. Related Works

2.1. Mobile Agent Description

In general, agents are useful because they understand some aspects of their user’s goals and can carry out actions autonomously to fulfill those goals. Mobile agent is a kind of independent program, which can migrate from one node to another node in distributed network by oneself, which is to say, in an autonomous way. The basic attributes of mobile agent are autonomy and mobility, which aim to debase network transmission and implement asynchronous interaction. Compared to traditional distributed techniques such as Client/Server model and Code On-Demand model, mobile agent has much advantage over them:

- can make proper use of existing resources so as to fulfill user's assignment
- can debase network traffic
- can balance network load
- fault-tolerance
- support mobile user
- support customized services

A mobile agent consists of three parts: resource, method and some auxiliary information. The resource portion can contain hardware resources, computing environment and original

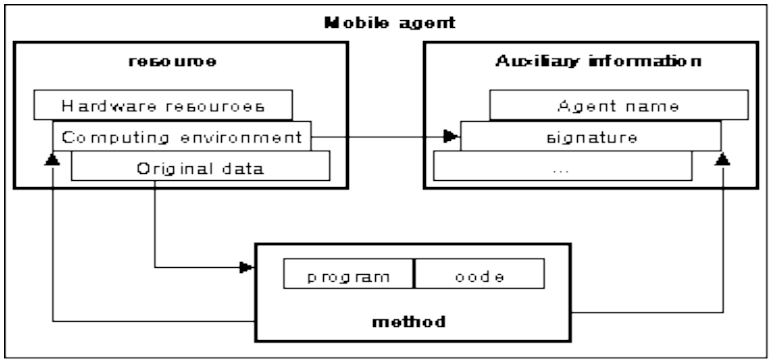

Figure 1. Constitution of Mobile Agent

data, computing environment means the serialized state of the agent encoded in the method part. The method part includes concrete computing code and program. The auxiliary information specifies descriptive attributes of agent including agent name, signature and *etc*. The structure of mobile agent is illustrated in Figure 1.

Mobile agent has its own life cycle including state of requirement capture, design, coding several testing, deployment and maintenance same as traditional software lifecycle. Figure 2 shows the life cycle of a mobile agent: In order to fulfill a task, mobile agents must have two essential factors: method and resource. If there is not enough resource or some kinds of service are absent, then migration is necessary.

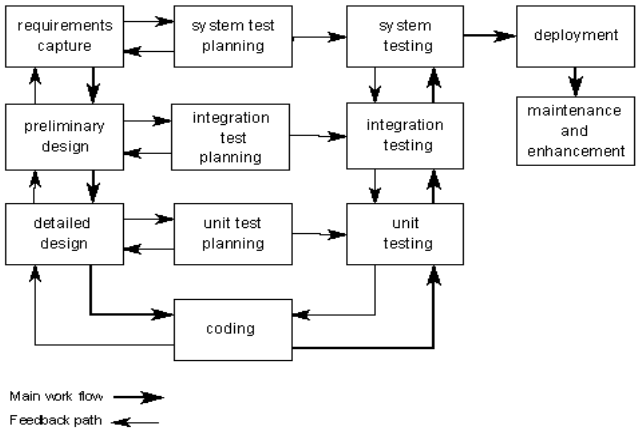

Figure 2. Mobile Agent Lifecycle

The role concept allows the part played by an agent to be separated logically from the identity of the agent itself. The distinction between role and agent is analogous to that between interface and class: a role describes the external characteristics of an agent in a particular context. An agent may be capable of playing several roles and multiple agents may be able to play the same role. Roles can also be used as indirect references to agents. This is useful in defining re-usable patterns. Resource is used to represent non-autonomous entities such as databases or external programs used by agents. Standard object-oriented concepts are adequate for modeling resources [5]. Figure 3 gives an informal agent-centric overview of how these concepts are inter-related.

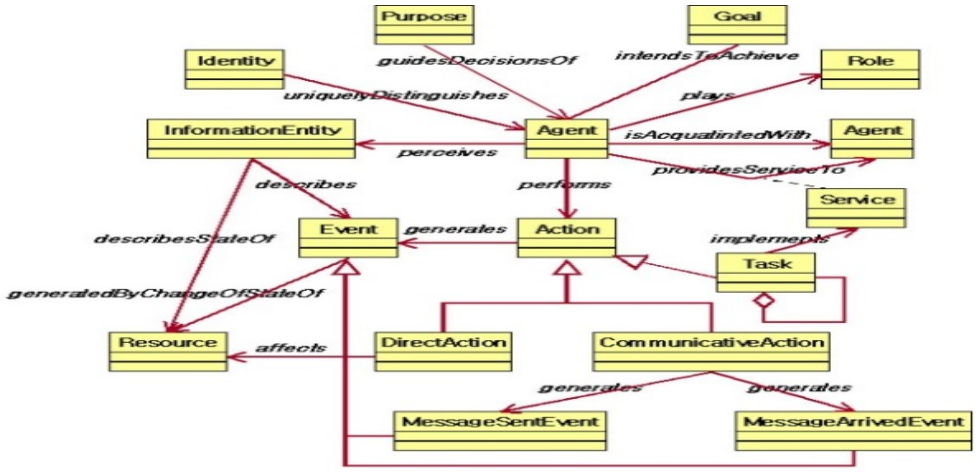

Figure 3. Agent Concept Model

2.2. Security Problems

Mobile agents offer a new paradigm for distributed computation, but the potential benefits must be weighed against the very real security threats they pose. As mobile agents can carry out actions autonomously, this may require that any given agent know personal or sensitive information about its user, and that it must be robust against revealing this information to third parties or allowing its actions to be subverted by a malicious interloper into carrying out an undesired action. For example, if there is no mechanism to prevent attacks, a host can implant its own tasks into an agent or modify the agent's state, this can lead in turn to theft of the agent's resources if it has to pay for the execution of tasks, or to loss of the agent's reputation if its state changes from one host to another in ways that alter its behavior in negative ways. Moreover, if mobile agents ultimately allow a broad range of users to access services offered by different and frequently competing organizations, then many applications will involve parties that may not trust each other entirely. There are two kinds of security issues specific to a mobile agent system:

- malicious agent: protection of the host against agent and protection of other agents
- malicious host: protection of the agent from the host and protection of the underlying network

Mobile agent is subject to the fundamental threats of disclosure, modification, denial of use, repudiation, misuse or abuse, replay and resource exhaustion which is briefly described as follows:

- disclosure: Information is revealed from monitored communications.
- modification: Communicated data item of agent is changed, deleted or substituted.
- denial of use: Users of mobile agents deny the actions done.
- repudiation: A party to a communication exchange later denies that the exchange took place.
- misuse or abuse: An entity pretends to be a different entity, for example, one agent pretends to be another.
- replay: A captured copy of a previously sent legitimate agent is retransmitted for illegitimate purposes.
- resource exhaustion: A resource is deliberately used so heavily that service to other users is disrupted.

2.3. Component Reference Architecture for Mobile Agent Development

In order to construct component reference architecture, agent is classified in general agent type and e-business function attribute. Figure 4 is a component and meta architecture based on the above described for mobile 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. These threats are possible not only when the agents travel but also when they are in their environment. A secure agent-based model must provide services to counter these threats by confining key security functionality to a trusted core that enforces the essential parts of the security policy.

Figure 4. CBD Reference Architecture of Mobile Agent

3. Convergence Applications Agent Component Development Process based on Architecture

As we suggested CBD reference architecture in previous section, component development process based architecture is a set of activities and associated results, which lead to the production of a component as in Figure 5. These may involve the development of component from UML model. In Figure 5, 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 systemically development process using AUML and design pattern technology to analysis, design, and develop mobile convergence agent. The domain analysis specification, design model, implemented component, which are produce in process, are stored in the repository [6].

3.1. 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 whether 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 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.

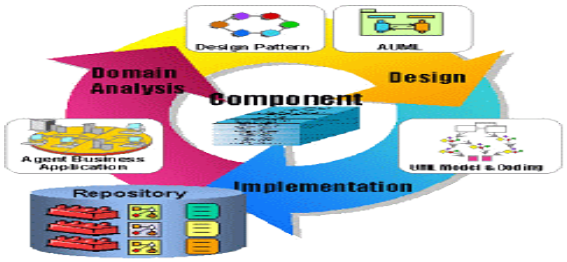

Figure 5. Component Development Process

3.2. Mobile Agent Design Phase

The mobile agent for convergence applications with adaptable component is designed based on domain requirement. Attribute and behavior is 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 UMLs dynamic model; sequence diagram, collaboration diagram, activity diagram 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 presets to use activity diagram and state charts.

Second, design pattern can be applied to previously identified area in reference architecture. We use mobile agent for convergence design pattern matrix for meta architecture of component reference architecture and design pattern. 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 be implemented though analysis and design phase, also possibly apply to entire lifecycle.

3.3 Mobile Agent Analysis

The dynamic design of the mobile agents model are required for mobile computing, it allow the wide-area collaboration. The dynamic creation of collaborative security enclaves

can ensure the correct application of each collaborating organization's security measures while also imposing any collaboration specific security measures.

The model can meet the requirement for collaborative environments:

- Multiple, dynamic user roles: the model allows users rights to be inferred from their roles. Moreover, it allows users to take multiple roles simultaneously and change these roles dynamically during different phases of collaboration.

- Flexibility: the model can support fine-grained subjects, objects and rights, that is, it allows independent specification of each right of each agent on each object. Thus it allows independent protection of each agent in a collaborative environment.

- Efficient storage and evaluation: the storage of security policy definitions and evaluation of the checking rule is efficient. Automation: the model makes it easy to implement secure control in multi-user applications.

Because an agent carries the security preferences by itself, it usually includes rules that govern its consent for cooperation. However, the agent has to rely on the context for compliance. The agent's security preferences describe whom and under which circumstances the context or another agent may dispose of deactivate, clone, dispatch, or retract the agent. The preferences may further define which other agents may call which of its methods.

It's important to note that sharing must be confined to the selected collaborative objects and to the intended participants when a collaborative work is developed especially over a distributed network or system. Thus the fine-grain security control scheme provided by the model is compatible with the requirement of the distributed environment, particularly in the area of collaboration.

3.4. Mobile Agent Model

We have developed a security model that provides an overall framework for agent security. The model supports the flexible definition of various security policies so as to meet the dynamic requirements of distributed system.

The model uses mobile agent technology to implement application specific security functions. The general architecture is shown in Figure 6.

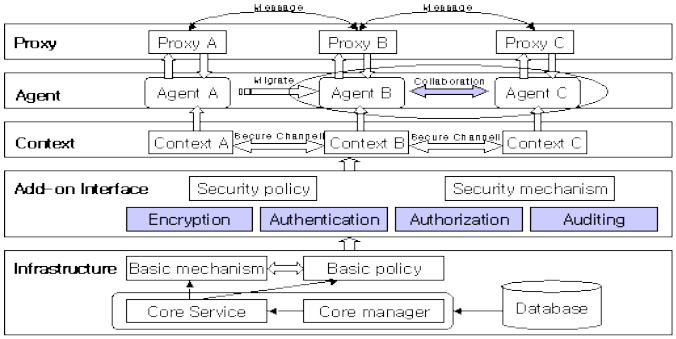

Figure 6. Mobile agent-based Security Model

- Infrastructure

The architecture is based on a security infrastructure that centered on a core manager. The manager acts like a reference monitor for distributed objects. To provide a secure base for this model, it is assumed that the architecture has a preconfigured core security service that

provides basic public key encryption and authentication. Local object invocations pass through the manager and it provides the core service. Logically the infrastructure has two major parts, security mechanisms and policies. These two parts interact with each other. A policy database represents the policies and mechanism.

The model supports reflection and permits run-time modification of policy and mechanisms. Such modification is the basis for adaptive security schemes that react to security attacks.

- Add-on Interface

Based on the core services, a layer of security interface is designed which can provide add-on security services such as encryption, authentication, authorization and auditing. Because these services which can be applied directly are separated from basic infrastructure, thus the model can provide flexible way to inject new security service and policy dynamically.

Security policies are defined in terms of a set of rules by one administrative authority. The policies specify the conditions under which agents may access objects, the authentication required of users and other principals which actions an authenticated entity is allowed to perform, and whether entities can delegate their rights are also included. The communication security required between agents and between corresponding contexts is another emphasis. In the end, the degree of accountability required for each security relevant activity must be took into account. An agent might be aware of the security policy of the hosting context and how it is enforced. If so, the user can be authenticated prior to creating the agent; security is then enforced automatically.

- Context

A context is an agent's workplace. It is a stationary object that provides a means for maintaining and managing active agents in a uniform execution environment where the host system is secured against malicious agents. The context manifests the execution platform of the agent. Its identity the URL of the host together with a qualifier if there is more than one context. Unlike agents, contexts are long-lived objects and thus may keep their identity that is, their address even after updates or complete replacements of the software and hardware that realize the context. For security-critical applications that associate trust with a specific version of the context, the identity of a context must have an attribute like the serial number of a CPU. The context must be responsible for its safety and security and its underlying machine by means of defining the security policy so as to protect local resources against agents. The context is also responsible for guaranteeing that no agent can interfere with any other agent active in the same context if not explicitly permitted by the corresponding agent.

- Agent

Agents are autonomous objects that visit agent-enabled hosts in a distributed network. An agent that executes on one host can halt execution, dispatch to a remote host, and resume execution there. When the agent migrates, it takes along its program code as well as its data. Every agent has an identifier that is unique over its lifetime and independent of the context it is executing in. The identifier might be used in policies that refer to specific instances of agents; for example, it might indicate that a particular agent could dispose of any of its offspring.

Agent can be created or derived by ancestor, the safety of agents must be kept by themselves in the way of defining terms of liability, security preferences and so on. These methods can limit the agent's capabilities. In this model, an agent is created within a context. The agent starts to execute as soon as it has been initialized. Dispatching an agent from one

context and insert it into the destination context, where it will continue its execution. Deactivation of an agent removes it temporarily from its current context and holds it in secondary storage. Activation of an agent restores it into a context. Disposal of an agent halts its current execution and removes it from its current context. Besides agent migration, agents from different context can collaborate in fine-grained scheme especially suitable for distributed environment.

- Proxy

A proxy is a representative of an agent. It serves as a shield to protect the agent from direct access to its public methods. The proxy also gives the agent location transparency thus it can hide the agent's real location. As agents exchange or communicate by means of message passing, and mobile agents do not exist in statically configured object structures, thus the most important function of proxy is that it allows flexible interaction and exchange of knowledge between agents.

The structure of the model permits reflection, that is, it permits the security operations to be changed dynamically. Applications of reflection in a security architecture include improved control of the system, counter- attacking security attacks by increasing surveillance, auditing, and security measures such as isolating, monitoring, spoofing compromised remote agent, and replacing compromised encryption or security algorithms.

3.4.1. Mobility for Mobile Agents

In order to ensure the migration mobile agent mobility, the model must be properly identified with the help of a suitable certification infrastructure. Before a user dispatch an agent, the context will authenticate the dispatcher's identity in the way of certification or signification so as to confirm that the dispatcher is a registered user. Within the creation request, the agent user defines security preferences to be applied on the agent. The preferences including the unique attributes of the agent together with the agent code and it will be signed by the context. Thus any receiving context can verify the integrity of the agent.

Agent moves when it is either dispatched to or retracted from a remote location. A secure channel is established between the context that receives an agent and the context that delivers the agent to the receiving context. The current context protects the integrity of agent data by computing a secure hash value that allows the destination context to perform after-the-fact detection of tampering during the agent's transit. Unauthorized parties can be prevented from reading sensitive information held by an agent while it is in transit between two agent contexts if the peer contexts agree on the use of cryptography for encryption.

For each context, a security policy describes the proper communication mechanism with any peer context. For example, although an agent might not require any security protection for its transfer to the destination context, the destination context's security policy may lay down the use of the SSL protocol with client authentication.

3.4.2. Execution Security

When an agent enters a context, the context receives a reference to it and the agent resumes execution in the new context. The agent can also obtain a reference to the context interface. An agent uses the context to gain information about its new environment, in particular about other agents currently active in the context in order to interact with some of them.

Contexts have to protect themselves against agents and agents must be precluded from interfering with each other. The agent context establishes a reference monitor, which gives an agent access to a resource only if it complies with the access control policy. It might give

permission to obtain file information; to connect to a network port on the origin context or to any other context; or to load a library. Thus a context establishes a domain of logically related services under a common security policy governing all agents in that context.

4. 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 systemically 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 mobile agent and agent-oriented system can obtain the efficiency through component reuse. In this paper, we present a mobile agent-based security model, which aims to solve the security problems. The model supports the flexible and extensible application specific security measures required by mobile computers and devices in distributed system. In particular, we gave a full description of mobile agent and its migration mode, after discussing the security thread to which mobile agent faces, the model is explained at length and security analysis is also given.

Acknowledgement

This study was supported by the Kyungil University Grant.

References

[1] S. Gerard, F. Terrier and Y. Tanguy, “Using the model paradigm for real-time systems development: Accord/uml”, Lecture notes in computer science, vol. 2426, **(2002)**, pp. 260-269.
[2] S. Cranefield and M. Purvis, “UML as an ontology modeling language”, Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), vol. 212, **(1999)**.
[3] K. Czarnecki and S. Helsen, “Classification of model transformation approaches”, Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, **(2003)**.
[4] C. J. Beckmann, “Horizons in Scientific Networking and Distributed Computing”, Computing in Science & Engineering, vol. 1, no. 1, **(1999)**.
[5] A. Fuggetta, “Understanding Code Mobility”, IEEE Trans. on Software Engineering, vol. 24, no. 5, **(1998)**.
[6] C. G. Harrison, D. M. Chess and A. Kershenbaum, “Mobile Agents: Are they a good idea?”, IBM Research Report.
[7] N. M. Karnik, “Design Issues in Mobile Agent Programming Systems”, IEEE Concurrency, vol. 6, **(1998)**.
[8] N. R. Jennings and M. Wooldridge, “Agent-Oriented Software Engineering”, Proceeding of IEA/AIE, 1999, **(1999)**, pp. 4-10.
[9] J. Odell, “Agent Technology”, OMG, green paper produced by the OMG Agent Working Group, **(2000)**.
[10] M. P. Papazoglou, “Agent-Oriented Technology in support of E-Business”, Communications of the ACM, vol. 44, no. 4, **(2001)**, pp. 71-77.
[11] J. Odell, H. Van Dyke Parunak and B. Bauer, “Extending UML for Agents”, Proceeding of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, **(2000)**.

[12] B. Bauer, J. P. Müller and J. Odell, "Agent UML: A Formalism for Specifying Multiagent Interaction", Proceeding of 2000 Agent-Oriented Software Engineering, **(2001)**, pp. 91-103.
[13] H. S. Nwana, "Software Agents: An Overview", 96 Software Agent Technologies, **(1996)**.
[14] M. L. Griss, G. Por, "Accelerating Development with Agent Components", IEEE Computer, vol. 34, no. 5, **(2001)**, pp. 37-43.
[15] P. Brereton and D. Budgen, "Component-Based Systems: A Classification of Issues", IEEE Computer, vol. 33, no. 11, **(2000)**.
[16] Y. Aridor and D. B. Lange, "Agent Design Patterns: Elements of Agent Application Design", Proceeding of Autonomous Agents, vol. 98, **(1998)**, pp. 108-115.
[17] G. T. Heineman and W. T. Councill, "Component-Based Software Engineering", Addison-Wesley, **(2001)**.