UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

Design Patterns to Enable Agent-based Mobile Service Application Development.

Sameh Abdel-Naby and Paolo Giorgini and Michael Weiss

December 2006

Technical Report # DIT-06-096

Design Patterns to Enable Agent-based Mobile Service Application Development

Sameh Abdel-Naby
University of Trento
Via sommarive, 14
38050 Povo (Trento), Italy
+39046188207
sameh@dit.unitn.it

Paolo Giorgini
University of Trento
Via sommarive, 14
38050 Povo (Trento), Italy
+390461882052
paolo.giorgini@unitn.it

Michael Weiss
Carleton University
Ottawa, Ontario
K1S 5B6, Canada
+1-613-520-2600
weiss@scs.carleton.ca

ABSTRACT
This paper proposes a new framework to develop agent-based mobile phones service application using the current constructions derived from agent patterns approaches. Agent patterns are categorized in technically various modules that interconnect according to the users' demands and application needs. Our framework, applying a multi-agent systems technique, allows users to drag and drop certain system plug-ins to enable explicit functions; this will shape the final behavior of agents within the application and determine its characteristics. We provide an example of a real-life application to illustrate the implementation of this framework.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence— *Multiagent systems, Intelligent Agents*; D.2.2 [**Design Tools and Techniques**] - *Object-oriented design methods*

General Terms
Management, Performance, Design, Reliability, Human Factors

Keywords
Agent, Multi-agent systems, mobile agent, design pattern, advanced communication methods.

1. INTRODUCTION
The development of mobile services is one of the most interesting research areas, and is directly related to the study of *Mobile Information Systems* (MIS) [1]. The recently developed and more sophisticated mobile operation architectures enabled many applications within the mobile-user side of the general structure. For example, the third generation of mobile communication systems (3G) [2] enabled the convergence of telecommunications and data communications industries. By extension, the convergence of mobile technologies and the Internet allows for persuasive possibilities for future applications and solutions, which shape and reflect on Mobile Service Frameworks. The present diversity of mobile service applications that consider both the surrounding environment and location characteristics (e.g., ToothAgent [3]) contribute to the process of convergence described above.

In an application implemented using agent-oriented programming techniques, which are located in a specific environment or a network, users expect this application to fulfill a set of requirements that are hardly realized if other techniques in use. The two major characteristics of *Software Agents* [4] are self-containment competence and the ability to interact properly with the surrounding software entities. The use of software agents can create an "intelligent" interface between users' preferences and the back-end systems. Agents are now able to interact and communicate with each other, forming a virtual community and feeding the user back with suggestions. MoPiDiG [5] and similar implementations involve the use of advanced communication methods such as Bluetooth and Wi-Fi in order to accomplish an overall architecture that finally deliver a certain service to a specific mobile-user in the shortest route possible. Re-using the same software entity in different applications is a repetitive scenario that is usually seen in delivering a complex application, and noticeably realized in developing an Agent-based application.

In this paper, we propose a framework for applying Design Patterns [6] as a method to enable the development of Agent-based mobile service application. This framework divides the mobile service application development process into two main phases. The first phase includes the integration of commonly used software entities within the concluding desired architecture and, which is achieved according to responsibilities and tasks delegated to a specific agent within the overall platform. The second phase involves the development of new software entities that are have not been previously applied yet and have the potential to integrate in other similar implementations.

The rest of this paper is structured as follows: Section 2 outlines our motivation. Section 3 explains the proposed framework in detail. Section 4 introduces the integration between Agent-based mobile services application and design patterns. Section 5 examines this framework by applying it to an example and presenting a possible simulation platform. Section 6 concludes the paper.

2. MOTIVATION

Making highly needed services available for mobile and computer users has been the subject for many research projects. Service Oriented Architectures (SOA) presented a new approach to address a range of commonly desired services to computer users. SOA sub-discipline, Mobile Service Oriented Architectures (MOSOA), further enhanced the perception of service delivery by targeting end-users and offering them the same and, even better services at their own portable devices (e.g., cellular phones or PDAs). Also, many ongoing research projects [7, 8, 9] aim to realize the optimal method to make mobile service oriented architectures more reliable and efficient. For example, a widely used technology like the *mobile java client* offers a standard operational model for traditional cellular phones. This technology has facilitated the integration between computer-based and mobile-based application development.

Telecommunications has recently seen a rapid increase in the number and quality of available services and applications. Adjustments taken to the so-called *Network Life* are now real, and the users' required tasks became simpler to make, although complex processes are used. Common *portable computers*, *PDAs* and mobile phones, have integrated advanced communication methods (e.g., Infrared, Wireless and Bluetooth) that facilitate the establishment of *Networked Life*. Also, these devices are now enabled to have many pre-installed and downloadable applications that interconnect with several communication objects and devices, which make these scenarios reach well-to-do tasks in a clear practical method. These developments changed the way scholars observe the need for Mobile Services to be integrated within other frameworks and architectures.

The focus of research on *Multiagent systems* [10] is the system design, in which a large number of intelligent agents interact together to achieve a set of pre-defined goals or complex tasks. These agents are independent software entities, and can be formed as software programs or subparts of them. Their interactions can be either joint or self-interested, depending on whether they share the same goals. Research project are concerned with developing agent-targeted communication languages, defining a set of protocols that make agents within a system able to interact suitably and, finally, creating agent architectures that facilitate the design of specific *Multiagent system*. Agent-based systems and mobile-based applications are two promising approaches in the direction of enhancing Service-Oriented Architecture (SOA). In addition, objects-identification related technologies helped the development of *Smart Ambient* systems that can interact with humans and objects to improve the surrounding atmosphere. The merit goes to *Multiagent Systems* in forming a virtual community between environmental objects. This virtual community is reasonably interactive, and it provides data that can be used forward in delivering a service. Moreover, it allows us to raise the idea of location-based services, which help mobile users achieve their demands in a shorter and smarter route. A situation, in which a mobile user in a specific area and his/her mobile device is recognized by all of the surrounding devices, increases the usability of a range of resources that may have never been employed before.

Developing a mobile application is both a demanding and challenging task for software designers. Design Patterns [6] theories provide a significant tool in software development for they allow development practices and solutions to be recognized thoroughly and assist the interaction among separate system design concerns. Patterns can be seen as a directorial repository, a method to record the gained design knowledge and enable its re-use in the later development processes. Therefore, patterns are adopted to improve structural design practices. Recently, patterns application has been carried on its precision to finally reach a specific field problem. In a new field like *Mobile Services*, appropriate patterns should be fulfilling the needs of mobile-application design.

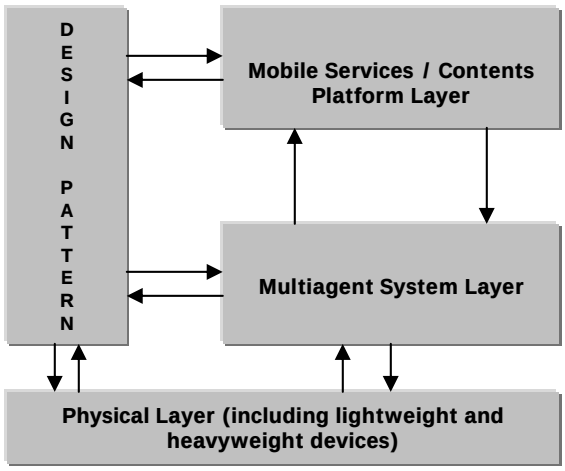

Figure 1. Mobile Services Design, Multiagent Systems and Design Patterns Cross-layers approach.

The above discussion indicates the possible integration between three components: 1) Mobile service application design architectures that are used to model the abstraction of delivering services to mobile-users, 2) Multiagent systems design architectures (with their contribution in increasing the usability of mobile-based service application), and 3) Design patterns, taking advantage of its ability to allow parts or the overall object-oriented programs to be reprocessed in a newly developed application (See Figure 1).

3. FRAMEWORK REPRESENTATION

3.1 Mobile Services Classification

For a platform to offer a set of services particularly these applied to portable devices (e.g., mobile phones, PDAs), it needs to obtain certain characteristics to produce final results to human users. These characteristics facilitate the integration between computer-like service application and the portable devices operating systems. It is not yet standardized application but it is almost realized that common mobile operating systems (e.g., Symbian-based mobile phones) allow explicit software unit to be reasonably customized to fit in the overall data flow of a mobile device. Many current research projects aim to standardize the mobile side operating systems in order to provide better end-users services. What remains is to standardize the mobile-based service software components and to make them in plug-ins modules, separate agents, which are able to interact, coordinate and finally

integrate to deliver a service, through forming a useful mobile application.

We classify service divisions that are commonly used in designing mobile-based applications. These services are made of software entities/agents, each of which is in charge of achieving sub-tasks of the overall service goal depending on what it was designed for (e.g., a communication agent, coordination agent or a negotiation agent). We made our service classification according to the literature feedbacks about frequently used applications (e.g., Tourists-targeted mobile applications and meeting/organizer kind of applications or directory services browsing mobile applications). Each of the core service classes is concerned with a specific problem domain, and, consequently, invokes a range of sub-classes related to the general interest of this specific class but in different solution modules.

There are four main mobile service application classes; each is responsible for three main tasks:

- **Communication Class:** A communication service application is responsible for external communication of the mobile-based application, with devices or other software entities in order to fulfill the user's demands and achieve the application goal. Another function is to make this communication work properly, through the communication methods used. The last function of this class is to allow the service application to reconnect with the collaborated actors, since some mobile applications need to launch certain offline tasks (e.g., server-side execution) in order to achieve certain goals.
- **Expression Class:** Each service offered to mobile users requires a certain data arrangement, depending on the type of services provided and the back-office running architecture (e.g., maps display and directory services). Therefore, a central function of this class application is the way in which data is organized in the client-side of the mobile service application. Another function is the means of interaction between different mobile users and the displayed data. The last function of this service application class balancing the displayed data on the overall service in terms of whether it requires detailed or superficial dataset execution.
- **Implementation Class:** Not all call methods of service-application to backend-system can be made on the mobile side, and vice-versa. This is essentially due to limitations on the mobile-side resources and the communication speed between mobile end-users and service operating servers. Hence, the implementation class is responsible for the organization of service tasks executed on the mobile side of the application, considering the mobile capabilities to perform such an operation. The second function of this class is handling the arrangements that make the service application server-side executing certain tasks that need special resources on behalf of the mobile-side application. The final function of this class is to monitor the utilized resources on the mobile side and to organize the overloaded tasks appropriately.
- **Combination Class:** Enabling mobile-based service applications to combine a range of different objectives is one powerful tool that a proper mechanism may add to certain architecture. Real-life scenarios suggest that the combination between two or more services or more enable mobile-user to achieve his/her goals sufficiently. For example, the combination between maps mobile display services and restaurant directory service would make a mobile user able to select a good restaurant and identify its location at the same time. The combination class is responsible for managing the service at this point from three different dimensions: 1) The server-side aspect that makes this class able to retrieve the required data from other linked distributed service servers, 2) The application-side aspect, responsible for merging the retrieved data and make it accessible by mobile user single interface, and 3) the client-side aspect that guides the pre-installed mobile application through the structure of collaborating servers.

3.2 Agents Services Communication

The above service classes are constructed to be independent from the specific Agents layer they are interrelating with. At the same time, however, they cannot be used without being installed in a proper agent profile. This usually occurs because they have no original means of communication and must be cross-layering with agents or multi-agents architectures to utilize the set of pre-defined communication protocols provided by them. An illustrative example for possible integration can be FIPA framework [11], which uses an Agent Communication Language (ACL) for communication. This communication language is located two layers below the mobile service application layer and on the same level as Agent Management and Agent Message Transport and can be extended to include Service Management. This allows the communication between the above service classes and their representation agents before the abstract architecture provided by FIPA.

Functions performed by Agents to represent service classes and their sub-classes are as follows:

- The first function of Agents is a search function to properly represent a specific service within a range of service categories previously defined by mobile-based service application developer. This function helps the agent to recognize the service that best fits into its predefined characteristics and facilitate the process of adaptation between service class and autonomous agent – together forming the Service Agent.
- The second function is to analyze the situated service class and, if available, its sub-classes. After the agent finds its representing service class, it returns to the *Agent Management* architecture with detailed description of this service. Thus a description analysis will be made by comparing the requirements and conditions drawn by the service with the characteristics given to an agent.
- The agent returns to the *Agent Management* layer with analysis results only if they are fitting. Consequently, saving these results will be the third function an agent has to take. Therefore, another sub-function is the waiting condition an agent should carry out in order to

give the possibility for the overall architecture to complete.

- At this point, the realized Multi-agent system consists of separate agents that are more shaped as general architecture plug-ins. These separate software entities will finally be integrated and examined for overall coordination and compatibility. The loop of functions can be repetitive once the plug-ins matching process produces any object failure. At this point, implementing an *Intelligent Expert Agent* mechanism is suggested, in order to prevent the system from replicating similar functions and, consequently, producing another object failure.

Furthermore, the Service Classification modules have to apply some system interaction rules:

- The first rule is to break-down the desired service class to as many subclasses as possible, and to represent each with its own detailed description. That is because the fewer the goals a service application is carrying out, the easier it is to look up for a proper agent to dress in, and, consequently, the clearer its expression.
- The second rule is to link the service class to an abstract hierarchy of Directory Services to help the agent locating the fitting classifications. A hierarchy approach is commonly used in Service Oriented Architectures (SOA) design, because it is easy to refer to a set of services with a general title, and this hierarchy is available whenever the demands increase to specify a certain service.
- The third rule is to attach a general service class to the architecture in order to simulate the data flow among the mobile-based service application and examine the proper delivery of service. This takes place once the function of integrating the selected agents and examining the general coordination of the multi-agent system components is completed.

The interaction between Service Class and an Agent has three prerequisites: 1) the availability of information about the set of controlling rules used to administer the external communication of the potable device, 2) the content architecture used and its development language, and 3) the enabling technologies used by the Mobile Operating System and the Mobile-based service application. Therefore, each processing phase must include a detailed specification of the supporting technologies and components of the portable device. Applying design patterns (discussed below) simplifies this task, as the experience obtained at each time the service class and the selected agent integrates, will be prospected for further implementations.

3.3 Applied Design Patterns Catalog

The iterative software engineering approach for multi-agent systems, which was first proposed by Lind [12], presented new ways to model the entire system, and captured several related aspects. Lind [12] has also presented a pattern catalog structure that we adopt in order to enable and enhance the development of agent-based mobile service application.

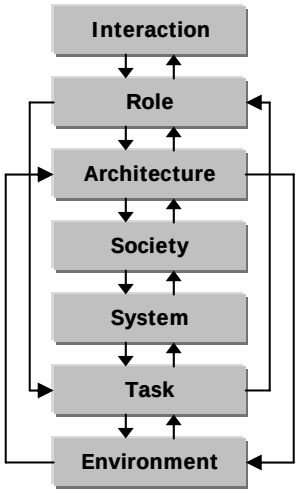

Figure 2. Applied Design Patterns Catalog Views.

Seven modifications to the views presented in this catalog structure are undertaken for the catalog to fit our framework (See Figure 3) as follows:

Interaction is a fundamental concept for a system that consists of multiple independent entities that coordinate with themselves in order to achieve their individual as well as their joint goals [12]. In this view, we emphasize the need for the developer to report the methods and techniques used to interact with the system during the design process. These interaction experiences will be recognized in further development scenarios and arise whenever similar situations occur. In designing a mobile-based application, certain restrictions (e.g. limited memory and processing resources) are applied and a unique interaction schema is expected. Multiagent systems that are implemented to represent a specific environment is quite complex and may apply several interaction in addition to the traditional one.

Role The role view determines the functional aggregation of the basic problem-solving capabilities according to the physical constraints of the target system [12]. In designing an Agent-based mobile service application, each agent is represented to solve a particular problem. Accordingly, the goal of each agent is to take the predefined path to resolve a problem. In certain scenarios, more than one agent may cooperate to resolve a more complex task, whereby each of the involved agents is sub-tasked to address a specific goal. This makes the role of the agent depends on the coordination protocol used to achieve the overall mobile service delivery.

Architecture (system, agent, agent management) The Architecture view is a projection of the target system onto the fundamental structural attributes with respect to the system design [12]. More often, several system integrations are made to a multi-agent system in order to reach to its ultimate goals. Particularly, integration commonly takes place between Agent-based mobile service application with other multi-agent systems, databases or web portals to enhance the quality of service provided to the end-user. In this view we outline the need to capture the full picture of

system integrations and sketching the general architecture, including lightweight and heavyweight devices integration and roles, in order to make it re-usable in applying these types of involvement on the long run.

Society is a structured collection of entities that pursue a common goal [12]. In a multi-agent system, separate software entities form the so-called virtual communities and communication between these communities even exceed the limitations of a single system and form the same kind of communities using a cross-networks approach. These entities usually share the same interests and goals and collaborate to obtain certain results. We devise a mechanism to control the creation of such communities. In particular, we apply the same kind of approaches in mobile service applications, emphasizing the condition of inserting a monitoring tool to observe the agents behavior in creating their own virtual communities. This monitoring tool will lead to a better prediction of society structure before implementing any of its parts.

System This view deals with systems aspects that affect several of the other views or even the system as a whole [12]. In a mobile-based service application, the System view handles the mobile user interface that relates to the interaction between the service application and the user. Covering all users' data-entry and portable device output functions, this view records the users' reactions in a certain situation and analyze it to develop a better and simple service module in case a problem occurs when the tries to operate a certain function.`

Task In this view, a task hierarchy is generated that is then used to determine the basic problem solving capabilities of the entities in the final system [12]. We adopt this view as it is but we suggest a link between this view and Role view. Such a link will enable the outputs (coming from the process of granting roles to agents in a mobile-based service) to be the inputs for creating a hierarchy for each agent capability.

Environment In this view, the environment of the target system is analyzed from the developer's perspective as well as from the systems perspective [12]. In designing an agent-based mobile service application, the developer's focus is on the way a service is implemented, tested and delivered. Whereas for a devise system, the focus is different since available resources and processing algorithms control the efficiency of the overall developed architecture, regardless of how no matter how sophisticated and maintained the systems are. Therefore, it is important to customize and integrate the perspectives of both the system and developers.

3.4 General Framework Composition

The composition phase of the general proposed framework involves the combination of the above-mentioned four mobile service classifications and the three delegated tasks of each. In addition to its designated functions, a *Service Agent* will operate to be compatible with its specific service class, in which Agents will apply a set of rules to ensure the proper match. To assist developers improve their design techniques, we suggest involving *Agent-Oriented Software Engineering* – taking advantage of the repetitive implementation scenarios – by applying *Design Patters Views* that will work on monitoring the experiences and notions obtained while designing an agent-based mobile service application (See Figure 3).

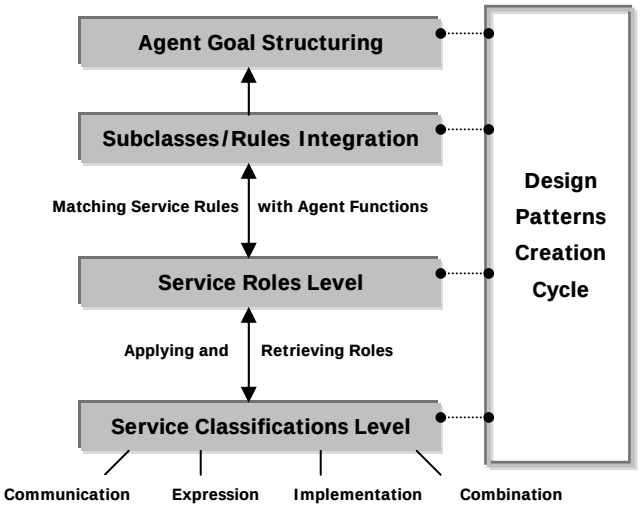

Figure 3. General Framework Composition Phase.

4. APPLICATION EXAMPLE

Based on the use of location and available car seats, Car-sharing systems allowed a substantial number of people to share car rides. These systems would, among other advantages, rationalize energy consumption, save money, and decrease traffic jams and human stress, and eventually make a significant improvement in human life. Car-sharing is a method to reduce the use of cars in a specific town or territory. Reducing car use helps, in turn, to decrease pollution and prevent some other problems. Car-sharing has three pre-conditions: 1) a car owner who uses his/her car to move from a place to another, and 2) another person who is interested to go somewhere along the car owner's path to destination, and 3) the willingness of the *ride seeker* to share the ride cost with the car owner.

Implementing a mobile-based service application that offers a ride-sharing service to university students responds to an increasing demand of students to commute between their universities and other destinations. Autonomous agents that take part of a Multi-agents environment for mobile based applications are well-known with their capabilities to achieve difficult organization and negotiation tasks within a certain community. Moreover, the ability of Agents to perform offline tasks on behalf of its user will gives system users the needed flexibility to move without being logged into the system. Therefore, integration between mobile application and multi-agent systems is possible and, accordingly, we suggest the use of autonomous agents in Car-sharing-like applications.

For a mobile-based application to communicate with distributed servers within the university infrastructure, it needs a means of communication protocol (e.g., Bluetooth, WiFi). These communication protocols are implemented in a system using new software entities. Therefore a system developer is driven to integrate extra design modules to facilitate the integration

between the multi-agent system and the methods of interactions/communication. The experiences that a developer obtain during the designing process to put together several and usually incompatible components, and the different interactions scenarios among system components and software entities, both alert us to the need to maintain *Design Patterns Repository* that record these experiences and facilitate the use of these documentations as references to further implementation activities.

5. RELATED WORK

An approach to Agent-based service composition and its application to mobile business process was presented by scholars [13]. They described an architecture model for multiagent systems that was developed in the European project LEAP (Lightweight Extensible Agent Platform). Its main feature is a set of generic services that are implemented independently of the agents and can be installed into the agents by the application developer in a flexible way. These generic services are responsible of the reusability of the common software entities and they handle most of the agent-related concerns (protocol, conversation, language, ontology, and errors), while allowing the developer to concentrate on the application logic. Another Agent System Development Method based on agent patterns was presented by other research group [14] and their method enabled developers to design process into two architectural levels and applying the appropriate agent patterns, and they added to the same method a higher level designs that are independent of specific agent platform so it can be reused.

6. CONCLUSION

This paper proposed a new framework using Design Patterns to develop Agent-based mobile service application. In the proposed framework, we introduced a classification of mobile-based application services using a multi-agent systems technique. We integrate this multi-agent system technique with a modified version of Lind's design patterns catalog to enable mobile phone users to drag and drop certain system plug-ins to enable explicit functions. By employing this framework in a real-life example, car-sharing systems, we demonstrated the implementation of this framework, which proves its reusability in other applications.

7. REFERENCES

[1] Taniar, D., Chao, H. C., Lee, D, L., Leong, H, V., Lin, B., and Peterson, L. L. The International Journal on Mobile Information System *IOS Press Trans. Volume 2, 2006*, ISSN 1574-017X.

[2] Trillium Digital Systems, Inc. Third Generation (3G) Wireless White Paper, *March 2002*.

[3] Bryl, Volha and Giorgini, Paolo and Fante, Stefano (2005). An Implemented Prototype of Bluetooth-Based Multi-Agent System. Proc. of WOA05, Camerino, *November 2005*. Also Technical Report DIT-05-062, Informatica e Telecomunicazioni, University of Trento.

[4] Hyacinth S. Nwana. Software Agents: *An Overview*. Knowledge Engineering Review, 1996.

[5] Seitz, C., Berger, M., Bauer B. "MoPiDiG" , Proceedings of the First International Workshop on Mobile Peer-to-Peer Computing, Orlando, Florida, USA, März 2004

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley.

[7] C. Carabelea and M. Berger. Agent negotiation in ad-hoc networks. In *Proceedings of the Ambient Intelligence Workshop at AAMAS'05 Conference, Utrecht, The Netherlands*, pages 5 - 16, 2005.

[8] C. Carabelea and O. Boissier. Multi-agent platforms on smart devices: Dream or reality? In *Proceedings of the Smart Objects Conference (SOC03), Grenoble, France*, pages 126 - 129, 2003.

[9] A. Rakotonirainy, S. W. Loke, and A. Zaslavsky. Multi-agent support for open mobile virtual communities. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI 2000) (Vol I), Las Vegas, Nevada, USA*, pages 127 - 133, 2000.

[10] M. Wooldridge. An Introduction to Multiagent Systems. John Wiley and Sons Ltd, February 2002.

[11] The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org, 2003.

[12] Lind, J. *Iterative Software Engineering for Multiagent Systems - The* MASSIVE *Method*, volume 1994 of *Lecture Notes in Computer Science*. Springer, May 2001.

[13] Berger, M. Bouzid, M. Buckland, M. Lee, H. Lhuillier, N. Olpp, D. Picault, J. Shepherdson, J. An Approach to Agent-Based Service Composition and Its Application to Mobile Business Processes Siemens AG, Muenchen, Germany; Mobile Computing, IEEE Transactions**,** Volume: 2, Issue: 3, 197- 206, July-Sept. 2003.

[14] Y. Tahara, A. Ohsuga, S. Honiden. *Agent System Development Method Based on Agent Patterns Proceedings of The Fourth International Symposium on Autonomous Decentralized Systems.* 1999.