*Papier soumis à IEEE Communications Magazine, special issue on Mobile Agents in Telecommunications*

# Enhancing Telecommunication Service Engineering with Mobile Agent Technology and Formal Methods

Marie-Pierre GERVAIS(*) and Alioune DIAGNE(**)
IUT Paris 5(*) - Université Paris 6(**)
Laboratoire d'Informatique de Paris 6 (LIP6) - France
{Marie-Pierre.Gervais, Alioune.Diagne}@lip6.fr
www-src.lip6.fr

## Abstract

In order to be competitive, telecommunication service providers need new technologies that facilitate the rapid introduction of validated services in a cost-effective manner. Service engineering is a new discipline in which the telecommunication sector adresses the technologies and engineering processes required for the service creation. Concurrently with these studies, the development of applications in Internet shows a new way to design telecommunication services based on the mobile agent paradigm. This brings new concepts that fit the requirements of service engineering. Therefore merging the approaches enables to improve the service creation process. We illustrate it by presenting a framework for the formal design of telecommunication services using the mobile agent technology complemented with formal methods. The contribution of this work is twofold. First, it relies on the RM-ODP and shows how to adapt its concepts to agent orientation. Second, it provides agent-based service designers with validation and verification tools to state about the quality of their specifications.

## 1. Introduction

Service engineering is an emerging discipline that appeared due to the telecommunication sector evolution. Because of the introduction of new technologies (e.g., ATM or wireless access) and the deregulation of the market, more and more attention is given to the services aspects. Actually, it becomes crucial for telecommunication providers to be competitive, and services are the means by which they can attract customers and thereby make some benefits. But this cannot be achieved without a real reduction of development duration and cost without affecting quality.

This implies that new technologies have to be used in order to facilitate the introduction of new services or the adaptation of the current ones. Then service engineering is the discipline that addresses the technologies and engineering processes required to define, design, implement, test, deploy, maintain and manage communication services [TRI 95]. Many projects have been initiated in this area by the telecommunication sector (e.g., ITU-T, ETSI, TINA-C or European projects such as RACE, ACTS or EURESCOM). All of them adopt a common approach that is the definition of an architecture and a methodological framework with its support. Thus the IN and TINA architectures have been defined. The methodological aspect is described by the Service Creation Environment (SCE) concept. This enables to unify the process of the service creation by defining a role model, a service life cycle model and a set of methods and tools that support the activities of all the roles.

Concurrently with these studies, the development of applications in Internet shows a new way to develop telecommunication services based on the mobile agent paradigm. A mobile agent is a computational entity that can migrate from one location to another in a heterogeneous environment. As an agent, it presents adaptation and interaction capabilities that provide the flexibility required by the service creation process. Most of the works related to mobile agents are concerned with the development of mobile agent systems that provide an environment for the agent execution. Some of them deal with the development of applications based on the mobile agent technology, mainly in telecommunication, mobile computing and mobile communication.

The convergence between these two above approaches begins to appear. The difficulty is first to integrate approaches and concepts from different areas such as software engineering for modeling and formal proof aspects, artifical intelligence for the agent aspects and distributed object computing for the architectural aspects. The second challenge is to apply them to the telecommunication area by taking into account its constraints. These are related in particular to the complexity, the scalability and the reliability of the system on which services are deployed. Services are themselves complex and large distributed applications. In that sense, their creation requires the expertise of the software development area in terms of methodology and life cycle. These provide methods that enable to structure the software architecture of the service and to prove it formally before considering operational aspects related to the implementation.

This paper presents a framework for the formal design of telecommunication services using the mobile agent technology. The work is related to a project in which we are associated with CNET. This project deals with the provision of methods and tools to a telecommunication services designer in order that he/she can formally specify the software architecture of a service as a set of cooperating mobile agents. Our approach is based on a software engineering process and is in conformance with the Reference Model of Open Distributed Processing (RM-ODP) standards developed by ISO and ITU-T [ISO 96]. The approach is devoted to the formal specification of the service using mobile agents. So far, we assume the existence of an agent system supporting the

This work is supported in part by grant 1B 95 from CNET (France Telecom Research Center)

service execution. In ODP terms, this means that we focus on the computational viewpoint rather than on the engineering one, as explained below.
The paper is organized as follows. We first review in Section 2 the requirements in telecommunication service engineering that motivate the use of the mobile agent technology. Then we present in Section 3 an overview of existing mobile agent systems and applications. In Section 4 we detail the framework that supports the formal design of services based on the mobile agent technology. We provide some concluding remarks in Section 5.

## 2. Requirements in Telecommunication Service Engineering

Requirements must be analyzed from two angles: the services customers and the services providers.
For the customers, the increasing number of services means that they have to face many services offering different functionalities, presenting information in different way and having their own operation. This implies the increase of the communication tasks and the amount of available information. Therefore, they need to be assisted to cope with this complexity. Another requirement is related to users mobility implying access to information from any location. But mobile communication results in reductions of traffic and connection duration. These constraints make the traditional client/server computing paradigm not suitable. Therefore an alternative model has to be provided to remove this drawback.
A service provider must well address the users requirements and be able to offer a rapid response. This can be customization of an existing service or development of a new one with minimal cost and time-to-market. This can be achieved by adopting the software components reuse approach enhanced with formal methods for quality assurance. To make this approach really efficient requires flexibility, adaptability and interoperability of the components as a service could be the result of a combination of components from different providers. Moreover, validation and verification based on formal methods must be supported in order to ensure the software quality.
The mobile software agent technology provides a solution that addresses these requirements. A mobile agent is defined in [GRE 97] as a software entity which exists in a software environment. It acts on behalf of other entities in an autonomous fashion, performs its actions with some level of proactivity and/or reactiveness and exhibits some level of the key attributes of learning, cooperation and mobility. So mobile agents offer the characteristics required by the combination activity of the service creation process. They can assist the user in his/her communication tasks. The agent mobility introduces the new computing paradigm of remote programming: the agent migrates to the computing resources and interacts locally with the server rather than calling remotely the service as in the client/server paradigm [WHI 95]. The number and the volume of messages between the client and the server is therefore strongly reduced, which is an advantage in case of mobile communications. An exhaustive study on the advantages of mobile agent technology can be found in [CHE 95].
However, this technology is not based on formal concepts that support the validation and verification required in service engineering. Consequently, using formal methods in addition to mobile agent technology enables to fully address the requirements of service engineering.

## 3. Overview of Mobile Agent Systems and Applications

Mobile agent technology models a network of computers as a collection of places acting as agent servers as they offer a service to the mobile agents that enter. Communicating applications are modeled as a collection of agents, each agent occupying different places at different times since it can move from one place to another. The technology has three major components: the langage in which the agents and places are programmed; an engine, or interpreter, for that language; and communication protocols that allows engines residing on different computers to exchange agents.
A large litterature describing mobile agent systems and applications has been published these last years. [GRE 97], [KIN 97] and [COC 97] are some examples of well-documented presentations of systems and applications from which this overview is extracted.

### 3.1. Mobile Agent Systems

As recently as three years ago, only a few mobile agent systems were under development based primarily on research languages like Tcl, Scheme and Oblique. Telescript or AgentTcl are examples of such developments [COC 97]. Among them, only Telescript from General Magic was commercially available. With the introduction of Java, a lot of mobile agent systems are now available and some of them are commercial systems (e.g., General Magic's Odyssey, IBM's Aglets, and ObjectSpace's Voyager) [KIN 97].
Mobile agent-based systems make use of concepts such as modularity to have structuration units, and interactions through well determined interfaces. Other concepts address the requirements coming with mobility security and transactions. For instance, mobility can be achieved through many different mechanisms according to the level of granularity attached to the structuration units (e.g., mobility can be at code operation level or at object one). Telescript contains explicit support for multiple agent transport mechanisms: Java Remote Method Invocation (RMI), CORBA's Internet Inter-ORB Protocol (IIOP) and Microsoft's Distributing Common Object Model (DCOM). Voyager has innovative migration as a virtual object can migrate while keeping its execution context [KIN 97]. Agent Tcl, developed by Robert S. Gray at Dartmouth College, has a docking system that lets an agent transparently jump off a partially connected computer and return later, even if the computer is connected only briefly [KOT 97].

Such a facility allows mobile agents to have interactions that are somewhat transparent to their location.

### 3.2. Applications

Mobile agent systems are becoming sufficiently well developed to be used as platforms supporting agent execution. Most of mobile agent based applications are telecommunication or mobile computing and communication oriented.
[KIN 97] reports that FTP Software uses agent technology for the automation of tasks in network configuration and management; Crystaliz designs a mobile agent framework to support the domain-distributed asynchronous collaboration with adaptative systems. The components in the systems, objects/documents, are agents that have a state and can migrate.
A similar approach is taken by AgentSys, a mobile agent system for digital media access and interaction on an Internet, developed at the University of Ottawa [FAL 97]. It takes the advantage of shifting from «viewing agents as mobile programs returning with media for documents» to «viewing documents themselves as mobile and filling themselves up».
MAGNA (Mobile AGeNt Architecture) is developed by GMD-Fokus to enhance the concepts of TINA and similar architectures (e.g. OMG's OMA architecture) towards a generic architecture for future telecommunication applications, in which the traditional client-server concept is harmoniously complemented by agent concepts. This architecture is the foundation for the longer term goal of developing a comprehensive framework for the developement of agent-based telecommunication applications which exhibit the aspect of rapid, decentralised provisioning of intelligent, partially multimedia-enabled services ("Intelligent services on demand") [KRA 97].

### 3.3. Conclusion

Mobile agent systems, especially those based on Java, share certain characteristics such as the production of an agent server as entry point on a given host, the migation from server to server and the load of the agent code from a variety of sources [KIN 97]. Today, many systems coexist and it is still not clear that there will be one «winner». Consequently different agent systems might have to work together. This leads to initiatives aiming to establish standards for mobile agent technology. The Federation for Intelligent Physical Agents (FIPA) and the Agent Society aim to produce such standards. Moreover, the Object Management Group (OMG) intends to define a Mobile Agent Facility (MAF) for CORBA in order to provide interoperability among agent systems.
Regarding the development of applications using the mobile agent technology, one remark is that authors do not aim to apply formal engineering methods in their applications structuration. Moreover, their applications architecture does not deal with separation of concerns required to cope with the inherent complexity of distributed applications such as telecommunication services. These need to be developed according to the methodological stages defined in service engineering through the SCE concept. Thus using mobile agent technology in service engineering needs to merge concepts and techniques from different areas.
Our contribution has two goals:

- define an agent model that fits the level of a generic structuration unit with a formalized basis that allows quality assurance (verification and validation);
- handle interactions between agents of an organization while taking into account assigned goals and their constraints on the achieved Quality of Service.

## 4. A Framework for the Formal Design of Telecommunication Services

### 4.1. Motivation

Our works aim to provide methods and tools to a telecommunication services designer in order that he/she can formally specify the software architecture of a service as a set of cooperating mobile agents. The approach is based on a software engineering process and is in conformance with RM-ODP. This is an architectural framework for the construction of distributed systems and applications. It provides an object model and an architectural model. The object model introduces a minimal set of modeling concepts. The architectural model defines the viewpoint concept. A viewpoint is a subdivision of a complex system specification. It corresponds to a particular perspective, allowing the system to be «viewed» from a particular angle, focusing on specific concerns. The five viewpoints are the enterprise, information, computational, engineering and technology viewpoints. The *enterprise viewpoint* focuses on roles, purposes and policies that govern the system. The *information viewpoint* focuses on information and information flows within the system. The *computational viewpoint* focuses on functional decomposition of the system into distributable units with well-defined interfaces. It is related to the software architecture of the system. The *engineering viewpoint* focuses on the structures and the components, i.e. the infrastructure necessary to support distributable objects. Typically, mobile agent systems as presented in Section 3.1 are such infrastructures. The *technology viewpoint* focuses on implementation choices and the realization of the system. The RM-ODP prescribes for each viewpoint a set of concepts and rules, called *viewpoint language*, each complying system must obey.
Although RM-ODP defines an architectural framework, it does not provide a prescriptive methodology that can be followed in developing a system. However, it is considered that a methodology can be based on the ODP viewpoints, each step being associated with an RM-ODP viewpoint [SALL 95]. This consideration is taken into account in our approach. Thus the requirement engineering is associated with the enterprise viewpoint, the service analysis with the information viewpoint, the service design with the computational viewpoint and the distribution and implementation description with the engineering and technology viewpoints.

We focus here on the part of our works devoted to the computational viewpoint which enables to specify a telecommunication service as a set of cooperating agents by means of an agent-oriented computational language. This must enable the specification of an agents combination resulting from the derivation and the refinement of agent and interaction-oriented design patterns. Such a combination requires the identification of agents that must be selected in a library of pre-defined agents. The selection is based on the external view the agent provides to the designer, i.e. its interfaces through which it interacts with other agents. Moreover, all the required agents may not exist in the library and may need to be developed for a specific service. So the computational language encompasses two aspects. At a coarse grain level, it must enable to put together pre-defined agents described by their interfaces, i.e. their interactions. Thus the language must support an interaction model. At a fine grain level, it must enable to define new agents and therefore to support an agent model. Finally, in order to ensure the quality, i.e. the robusteness and the reliability of the service, we make use of formal methods, not provided by the agent technology. Therefore, the language must support automatic translation into a formalism that enables validation and verification activities. We choose the Petri nets formalism because it is well-suited to reactive and distributed systems such as telecommunication systems. It is also richer than others generally used in the telecommunication area (e.g., SDL or Estelle) since its provides proof techniques based on mathematical theory.
The reminder of this section describes the agent-oriented computational language AF-Class we defined.

### 4.2. The AF-Class Language

The AF-Class language is composed of a specification model and a verification model. The specification model encompasses the agent and interaction models that enable to specify the software architecture of a service in terms of interacting agents. Once this computational specification established, then it can be validated through the verification model. This section details these models.

#### *The Agent Model*

An agent is first of all an entity that manages resources and has goals to fulfil. These goals can be induced by its own needs or derived from the contracts the agent commits itself to fulfil. For the own goals, agents have triggers which model behaviors automatically executed whenever the agent reaches some specified state. Triggers allow the agents to run autonomous behavior along with their interactions. Triggers are very important towards the achievement of agent oriented behaviors because they extend the request/reply based interactions between agents. To put control on the interactions and avoid faults propagation, an agent can have exceptions. Exceptions are executed whenever the agent considers that the information coming from its environment (mainly results of operations) does not match specified requirements. Then the agent runs the exceptions that allow it to protect itself from what is considered to be erroneous behavior or results. Exceptions are also the means enabling to model the agent mobility. Basically, the behavior of an agent is modelled according to the environment in which the agent is located. When the environment does not provide anymore the expected requirements, then the agent moves to a more appropriate environment, this action being modelled as an exception.

#### *The Interaction Model*

A computational specification is the description of the reactive aspects of an organization of interacting agents. As we deal with the validation of this combination, we focus on the formalization of the interactions between the agents and on the reactive capabilities supporting them. So we identify some interactions patterns that can be formally proved. These are based on a classification of interactions we propose and that is built upon the RM-ODP interactions. We first present this classification and then we describe the structuration rules that enable to build high level interactions from basic ones.

*Interaction Classification*

Our classification encompasses two groups of interactions: point-to-point interactions and multi-point interactions.

- The *point-to-point interactions* take place between two agents and act upon their current states without modifying their goals and plans. An agent knows the identity of its correspondent. This group of interactions includes the ODP interactions, namely the operations, signals and flows, and the information Dispersion/Retrieval interactions that enable to collect or to dispatch information. These interactions allow an agent to build or modify the perception it has on its environment or to serve/require elementary requests agreed upon elsewhere.
- The *multi-point interactions* take place between several agents. They can modify the goals and plans of agents according to the reaction of the others. They encompass the procurement interactions through which several agents attempt to bid for goods or services offered by an auctioneer, and the negociation interactions as described in [ROS 94]. Agents need to cooperate in order to satisfy their goals that could be conflicting ones. So the further behavior of an agent is dependent on the results of its previous interactions.

*Interaction Structuration Rules*

The classification of interactions enable to identify some interaction patterns modelled as high level interactions built from ODP ones, namely operation, flow and signal. For this, two levels of structuration are provided.
At the first level, operations are grouped into services that are exported through the agent interface [DIA 96]. Services are set of operations with some additional semantics. Actually, in addition to this ODP concept of operational interface, constraints on the

invocations sequence are expressed as a usage pattern of an interface. Such a sequence allows the agent exporting the service to enforce the other agents to avoid violating integrity constraints on the local resources. Any agent which wants to import a service must apply the usage pattern of the corresponding interface. Moreover an access semantic is described in terms of synchronous or asynchronous access (i.e. blocking, resp. non-blocking invocation). The non-operational interfaces, i.e. signal and flows interactions are not submitted to usage patterns. Signals allow agents to notify to each other some elementary events that occur. Flows allow them to exchange structured data through streams.

Up to this point, the interactions between agents are supported by services, signals and flows. The behavior of an agent is therefore the sum of its interactions with the other agents and its triggers. Though, it is strongly depending on what happens in the environment. Each agent has to achieve its own goals. The means to achieve a goal are the own capabilities of the agent eventually extended with the offers from the other agents which commit themselves to cooperate to this purpose. This set of agents is an organization cooperating to achieve the goal. We do not adress neither how to form and dissolve such organizations nor their life-times. The corresponding set of capabilities must be used contextually according to the current goal and an agent should be able to change its plan according to the results of the achieved services and/or the quality. For such an organization of agents aiming a common goal, the language must offer the ability to express dependences at the level of the organization. It needs therefore to afford the means to handle at a higher level this dependence on the environment. It is the purpose of the high level interactions.

Thus at the second level of structuration, a high level interaction is defined as *a set of services attached with quality of service expectations and sequencing rules according to the achievement of that quality.*

This notion of interactions enables to build plans that involve many agents and take into account their dependence towards their environment. Interactions allow to build organizations as sets of agents assigned with goals. Agents in an organization cooperate to achieve the goals. Goals can be specified beyond the level of an isolated agent even if it corresponds to the satisfaction of the desire of that given agent. In this last case, the organization corresponds to the agents the one with the desire must interact with in order to satisfy it. Furthermore, interactions allow to build new or elaborate services by combining pre-existing validated and verified services or agents. This enhances the reduction of the time-to-market when building new services.

*An Example of High-level Interaction: the Contract Monitoring*

A contract monitoring takes place in the context of the electronic market of services that involves:

1) the *secondary server(s)* that provide services through a network and

2) the *primary server* that is an «agency» server that acts as intermediary between the customer and secondary servers.

The primary server must fulfil the requirements of its customer in terms of Quality of Service (QoS) expressed through a contract between the customer and the primary server. This contract is decomposed into subcontracts delegated to secondary servers. Then the primary server guarantees the QoS of the contract based on the QoS that the secondary servers guarantee through their subcontracts.

The contract monitoring aims to supervise this distribution of commitments. For this, one observer is located on each secondary server to supervise the local subcontract. A coordinator observer is located on the primary server to manage the whole contract. Events affecting the QoS on one secondary server is either solved locally or reported to the coordinator. Observers are assigned with priorities that determine the side effects of their local solution on each other. Events reported to the coordinator are causally ordered according to a logical clock.

The contract monitoring can be thought as a high-level interaction that takes place in an organization of three kinds of agents: the primary server, the secondary servers and the observers, as illustrated in Figure 1.

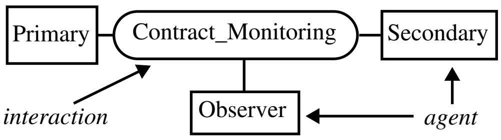


*Figure 1 - AF-Class Graphical Notation of the Organization*

Detailing the interaction shows the imported and exported services and the sequencing rules (Figure 2). Thus from the *init* point, an observer interacts with its secondary server to supervise its subcontract, i.e. it subscribes to some events (*Subscription* service) and it collects notifications on these events (*Obs_Announce* service). Once it gets a notification, it looks for a new local solution by using the *BD_req*. service. If it finds a new solution, it modifies its subcontract and notifies the primary server (*Contract_modif.* service) that forwards this notification to the concerned secondary servers (*PRI_Announce* service). If it does not, it notifies the primary server that a new global solution must be found (*Global_comput.* service). Before looking for a new global solution, the observers must be synchronized by the primary server (*Synchro*. service). The primary server can also kill an observer (*Destruction* service).

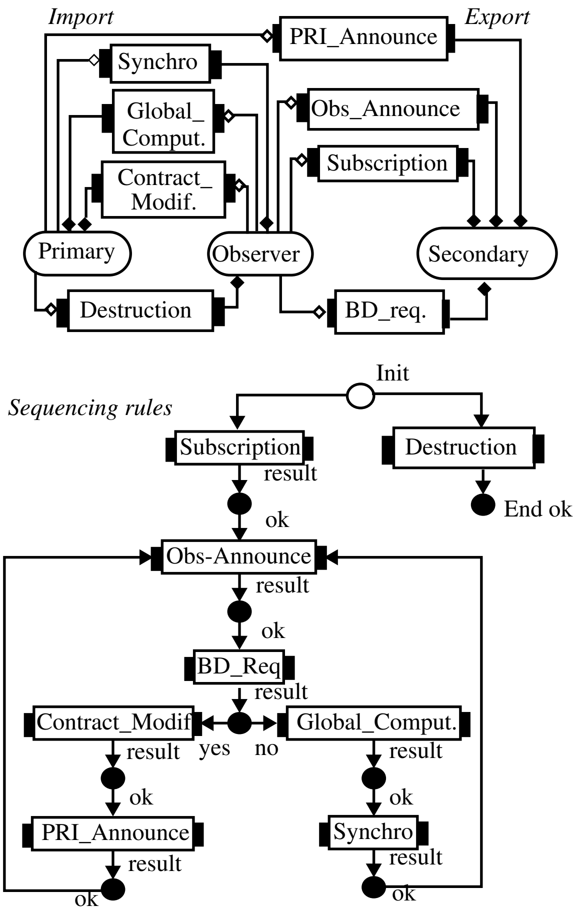


*Figure 2 - AF-Class Graphical Notation of the Interaction Structuration Rules*

***The Verification Model***

The verification model is obtained from the specification model by automatic transformation. It is expressed in the colored Petri nets formalism. One modular colored Petri net is derived from each agent model and a reachability graph is computed from it. This graph supports the model checking of Linear Temporal Logic properties. Some basic properties are always checked because they are strong conditions to the correctness of a specification. These properties are :

- dealocking states in which an agent is no longer able to support the interactions with its environment or its autonomous behavior,
- divergent states in which an agent loops forever executing autonomous actions and therefore can no longer support interactions with its environment,
- safety and reliability properties meaning that agents ensure results for the actions they have committed themselves to achieve and thus they are truthworthy.

Other properties depending on the application domain can be verified as long as they are concerned with states of agents, their behavior and they can be expressed with temporal logic semantics. Our software environment CPN-AMI[1] provides a set of tools that automates these verification tasks.

The verification enables to state about the achievement of the goal of a given organization. Given hypotheses on the environment not included in the model, it is possible to check if the organization is able to achieve its goal(s) or not. Hypotheses on the environment alleviate the uncertainty on it and the proof that can be given is still valid whenever the environment of the organization ensures these postulates. Hypotheses for which the proof fails give characterizations for the environment in which the organization is not viable. This is of help when complex organizations are built from other organizations.

*Example of the Contract Monitoring Interaction*

The verification model is derived automatically from the specification model described above. It is not represented for sake of place but some information on this model is provided in Table 1.

1. available at www-src.lip6.fr/cpn-ami

The analysis of this model enables to prove the following control properties:

- Basic control properties like safety and reliability of the agents which mean they neither come into blocking states nor behave in a confusing manner for the environment (requests are correctly processed and results are issued for them),
- Advanced control properties regarding the goals of contract monitoring. We have proved that an observer for which QoS is not guaranteed is always able to find a local solution or to ask the coordinator for a global one. The synchronisation of the agents for a global solution has been proved correct given the reliability of the message transmission.

The contract monitoring interaction then constitutes an interaction pattern that can be reused by any application which needs such a service, e.g., an electronic travel agency, an estate agency or anything else.

**Table 1 : The Verification Model**

| | places | transitions | arcs |
|---|---|---|---|
| **Observer** | 68 | 44 | 153 |
| **Secondary** | 45 | 28 | 114 |
| **Primary** | 47 | 31 | 141 |

## 5. Conclusion

Mobile agent technology brings new concepts that fit with the requirements of service engineering. Up to now, the main works are carried out in the field of agent-based systems and applications. But the specification of such systems and applications needs to be addressed. This paper is a contribution to the formalization of the agent-based telecommunication service creation. The contribution of this work is twofold. First, it relies on the ODP standards and therefore shows a way to formalize the concepts coming herewith and to adapt them to agent orientation (e.g., the interaction model). Second, it provides agent-based service designers with tools to state about the quality of their specifications. Services designed in this way can be deployed with a level of confidence beyond the one attained by testing in the classical way. Further studies are related to the adaptation of the other ODP viewpoints to agent technology. Thus, the concepts of the entreprise viewpoint are being investigated to check their relevance for agent organizations.

## 6. References


[CHE 95] D. Chess et al, *Mobile Agents : Are they a good idea ?*, Technical Report, IBM T.J. Watson Research Center, N.Y., March 1995

[COC 97] W.T. Cockayne and M. Zyda, *Mobile Agents*, Manning Editions, 312 pages, 1997

[DIA 96] A. Diagne and P. Estraillier, *Formal Specification and Design of Distributed Systems*, in Proc. of FMOODS'96, Paris, France, March 1996, pp341-356

[FAL 97] B. Falchuck and A. Karmouch, *AgentSys: A Mobile Agent System for Digital Media Acces and Interaction on an Internet*, in Proc. of the IEEE Globecom'97, Phoenix, USA, November 1997, vol. III, pp1876-1880

[GRE 97] S. Green et al., *Software Agents : A Review*, IAG (Trinity College Dublin and Broadcom Eireann Research Ltd), May 1997, www.broadcom.ie/~fs/agents.html

[ISO 96] ISO/IEC IS 10746-1 — ITU-T Rec. X901, *ODP Reference Model Part 1, Overview and Guide to Use*, 1996

[KIN 97] J. Kiniry and D. Zimmerman, *A Hands-on Look at Java Mobile Agents*, IEEE Internet Computing, 1(4):21-30, July-August 1997

[KOT 97] D. Kotz et al., Agent *TCL: Targeting the Needs of Mobile Computers*, IEEE Internet Computing, 1(4):58-67, July-August 1997

[KRA 96] S. Krause and T. Magedanz, *Mobile Service Agents enabling "Intelligence on Demand in Telecommunications"*, in Proc. of the IEEE Globecom'96, London, UK, November 1996

[ROS 94] J. Rosenschein and G. Zlotkin, *Designing Conventions for Automated Negociation*, AI Magazine, 1994

[SALL 95] J. Sallros, *TINA-C Service Design Guidelines*, TP_JS_001_0.1_95, March 1995

[TRI 95] S. Trigila et al., *Service architectures and service creation for integrated broadband communications*, Computer Communications, 18(11):838-848, 1995