*This paper will be published in the proceedings of Globecom'98 - November 1998*

# BUILDING TELECOMMUNICATIONS SERVICES AS QUALITATIVE MULTI-AGENT SYSTEMS: THE ODAC PROJECT

Alioune DIAGNE(*) and Marie-Pierre GERVAIS(**)
Laboratoire d'Informatique de Paris 6 (LIP6)
Université Paris 6(*) - IUT Paris 5(**)
4, place Jussieu - F75252 Paris Cedex 05
{Alioune.Diagne, Marie-Pierre.Gervais}@lip6.fr

## Abstract

The ODAC project aims to define methods and tools based on a formal approach so that a designer of telecommunications services can specify and implement a new service in the form of a Multi-Agent System (MAS). Our goal is to supplement the current approaches of the telecommunications industry by the use of the agent paradigm enhanced with formal methods.

In this paper, we focus on the works currently achieved in the project. These are related firstly to a service-based architecture of an agent-based system and secondly to our proposal for an Agent-Oriented Computational language (AOC language). Both are based on the concepts of the Reference Model of Open Distributed Processing standard (RM-ODP).

## 1. Introduction

The market of the telecommunications services is in full expansion and requires from the stakeholders an increased competitiveness, in particular due to the deregulation. The telecommunications services providers must thus be able to react rapidly to offer services corresponding to the demand of the various consumers. This requirement can be reached only by the control of efficient tools that help the introduction of new services.

This problem of the telecommunications services creation, or Service Engineering, is the subject of many projects in the telecommunications community (e.g., ITU-T, ETSI, TINA-C, RACE, ACTS or EURESCOM). The common approach adopted is first the definition of an architecture and secondly the definition of a methodological framework with its support [TRI 95]. Thus, Intelligent Network and TINA architectures were defined, respectively by ITU-T and TINA-C. The methodological aspect is described by the Service Creation Evironment (SCE) concept [SCORE]. This enables unifying the process of service creation by defining a role model, a service lifecycle model and a set of methods and tools that support the activities of all the roles.

Concurrent with these studies, the development of applications in the Internet shows a new way for the distributed applications design based on the agent paradigm. This presents adaptation and interaction capabilities that provide the flexibility required for the service creation process. The electronic trade, the search for information, the network management or co-operative work are well-known examples of applications developed with the agent technology. The design step of such applications remains however very empirical. Thus many research works relate to the aspects of methodology, modeling and formalization [WOO 95]. An integration of the key concepts of agent technology appeared essential that aims to produce specifications intended for industry. Therefore the FIPA[1] (Foundation for Intelligent Physical Agents) objective is to promote the development and the specification of agent technology. In the same way, the OMG[2] (Object Management Group) extended the scope of its work initially based on the object paradigm to integrate mobile agent technology in order to produce specifications.

Today, the search for a convergence between the works of the telecommunications and agent communities begins to become apparent. The difficulty is first to integrate approaches and concepts from distinct areas such as software engineering for modeling and formal proof aspects, artificial intelligence for the agent aspects and distributed object computing for the architectural

1. FIPA specifications are available at www.cselt.it/fipa
2. OMG specifications are available at www.omg.org

This work is supported in part by grant # TL97169 from CNRS (French National Center for Scientific Research)

aspects. Secondly, results of this integration must be applied to the telecommunications area by taking into account its constraints.

LIP6 works on this integration of concepts through its involvement in several national projects such as CARISMA with CNET (France Telecom Research Center) [GER 97a] and ODAC[1] with CNRS (French National Center for Scientific Research).

This paper presents the ODAC project. In Section 2, we draw the scope of this project. Section 3 and 4 focus on the works currently achieved in the project, namely the definition of a service-based architecture of an agent-based system and the proposal for an Agent-Oriented Computational language (AOC language). The architecture is defined according to an adaptation of the ODP viewpoints to the agent paradigm. The language is a formalism compliant to the ODP computational language concepts. It enables a designer to describe the specification of a telecommunications service during the design stage and it supports the formal validation of this specification. Section 5 provides some concluding remarks.

## 2. The ODAC Project

The objective of the ODAC project is to define methods and tools based on a formal approach so that a designer of telecommunications services can specify and implement a new service in the form of a Multi-Agent System (MAS). Our goal is to supplement the current approaches of the telecommunications industry by the use of the agent paradigm enhanced with formal methods. This brings various assets for the development of new telecommunications services such flexibility, high-level interactions or mobility. This approach makes it possible to consider the development of a telecommunications service as a MAS what offers a better support for the control and management distribution in telecommunications systems. However, it misses formal support to be fully efficient. Therefore we use the formal methods which bring the tools necessary to support the proof that guarantees the robustness and the quality of the applications.

To carry out this objective, we take into account the following elements:

- the agent metaphor as the computer-oriented view of an everyday life agent. Thus an agent is a software entity with characteristics of customization, delegation, contract monitoring and services combination;
- the Open Distributed Processing (ODP) standard, developed jointly by the ISO and the ITU-T, provides a reference model which defines an architectural framework for the construction of distributed systems and applications [ISO 96]. It defines the viewpoint concept as a subdivision of a complex system specification. Five viewpoints are identified: the enterprise, information, computational, engineering and technology viewpoints. The RM-ODP prescribes for each viewpoint a set of concepts and rules, called *viewpoint language*, each complying system must obey.

We provide an ODP-like structuration to agent-based services within open distributed systems in order to separate the concerns in a way which enables formal engineering methods.

Our approach aims at defining formalisms (i.e. language in the ODP terminology) which allow to specify a telecommunications service according to various levels of abstraction reflecting ODP viewpoints. The languages are mapped to a formal model (namely Petri nets) to enable at least simulation (e.g. the enterprise language) and verification of properties when it is possible (e.g. the computational language).

The awaited result of the project is the proposal for a model of specification in conformity with the ODP architecture which will include interaction patterns and which will support the formal verification. The model will be supported by an operational environment of formal design of MAS. This environment will provide design tools which will make it possible to a designer to specify a service by identification and instanciation of the interaction patterns and by their composition. It will also offer tools of proof making it possible to the designer to validate and verify his/her specification.

In this paper, we focus on the works currently achieved in the project. These are related firstly to a service-based architecture of an agent-based system and secondly to our proposal for an Agent-Oriented Computational language (AOC language).

1. ODAC stands for Open Distributed Application Construction

## 3. Service-based Architecture of an Agent-based System

Agent and Service are two valuable concepts to structure open distributed systems. They offer two levels of structuration which can be mixed to have a federative basis for such systems. Services can stand for structuration unit for which agents are used to represent their many concerns with a clear separation between them.

In order to issue an architectural proposal for structuring open distributed systems according to these concepts, we first propose to adapt the ODP viewpoints to the agent paradigm [DIA 97].

### 3.1. Adapting ODP viewpoints for construction of MAS

Agent paradigm is expected to emphasize in emerging open distributed systems the concept of service. This must take into account many aspects that can be divided into three parts:

- the *collaborative/cooperative aspects* like negotiating and making contracts with users or other services. These contracts can be commitments on quality of service and/or access rights, consequent billing, etc.
- the *cognitive aspects* like making inference on the contextual knowledge attached to a service or to its execution. This contextual knowledge can determine the way the service is offered. It can also be associated to the profile of the service user,
- the *reactive* or *computational aspects* like modifying the resources and running some specific processing necessary to exhibit the right functionalities under the contractual constraints. Services are namely reactive because they must maintain a continuous interaction with their environment (users or other services).

These previous aspects are not independent from each other. We can notice that the contextual knowledge used in cognitive activities may depend on the previous collaborative/cooperative activities and may influence the reactive ones. These interdependencies must be considered while using formal methods in order to support verification and validation. It appears therefore necessary to separate and manage the many concerns in order to avoid undesirable influences from each other [HUR 95].

As agent-based systems need more architectural guidelines to achieve a first level of integration as well as a good separation of concerns, we propose to proceed like in the RM-ODP with some adaptations [MKA 96]. The ODP viewpoints are well-suited to separate the many concerns in object-based open distributed systems. Nevertheless, we would not propose viewpoints for agent-based systems to be a progressive structuration like in ODP. We try, through our adaptation of viewpoints, to separate and organize the many concerns (collaborative/ cooperative, cognitive and reactive) of such systems and to make their mutual dependencies more manageable (Figure 1).

Adapting the enterprise viewpoint, a Service Manager Agent is an entity managing the policies and rules attached to the availability and utilization of a collection of services. A Service Manager must be able to negotiate the offer of its services with consumers. It must also be able to cooperate with other Service Managers to use their services when needed or to share the load of the service offer. Concerning the collaborative aspects, Service Managers must be able to establish contracts and to fullfil the subsequent obligations [PIT 95]. Service Managers must be able to accept or deny results of negotiation but once accepted, the subsequent contract must be carried out in a satisfactory way for the counterpart.

Adapting the information viewpoint, a Resource Manager Agent is an entity responsible to manage one or many resources on behalf of a Service Manager. It offers capabilities for access and modification of the managed resources. It defines the allowed access to the managed resources as well as integrity constraints that will be enforced. Resource Managers are under control of a given Service Manager which can give - by authentication means - an access/modification permission to other entities (any other kind of agent) [THI 95]. Exception processing must be enforced at this level to send some events back to the Service Managers because they might need some cognitive or collaborative actions. This allows to offer adaptable services. Exceptions allow one to have some kind of fault tolerance on Resource Managers.

Adapting the computational viewpoint, an Activity Manager Agent is an entity able to perform a set of actions in order to fullfil a given goal. An Activity Manager is under control of a Service Manager and receives from it access permissions on its attached

Resource Managers. It can also - according to its fixed goal - receive collaborative/cooperative and cognitive capabilities to address others Service Managers. Another possibility which seems best is that the responsible Service Manager carry on the collaborative/cooperative and cognitive activities with other Service Managers, then the Activity Manager only receives delegation on permission granted to its responsible Service Managers to access remote resources. Activity Managers may divide their goals into sub-goals. Therefore, they clone themselves into others Activity Managers to handle these sub-goals. The global coherence must then be managed by the agent which initiates the goal splitting. Transaction-oriented facilities must be supported to manage this coherence. Activity Managers can have collaborative, cognitive and reactive aspects according to their assigned goal. They are anyway attached to a given Service Manager which will delegate them part or whole of its collaborative/cooperative and cognitive capabilities.

The two last viewpoints in the RM-ODP deal mostly with implementation aspects. In the agent paradigm also, we will consider such kind of agents as relevant to realize the three previous classes we have defined. Depending on an underlining technology, one must consider how the previous levels of abstraction can be realized. What can be called *engineering* and *technological agents* must therefore be defined to establish some correspondence between the needs in the previous levels with the concepts available in the underlining technology. We remain deliberately unprescriptive and refer to the RM-ODP for adaptation.

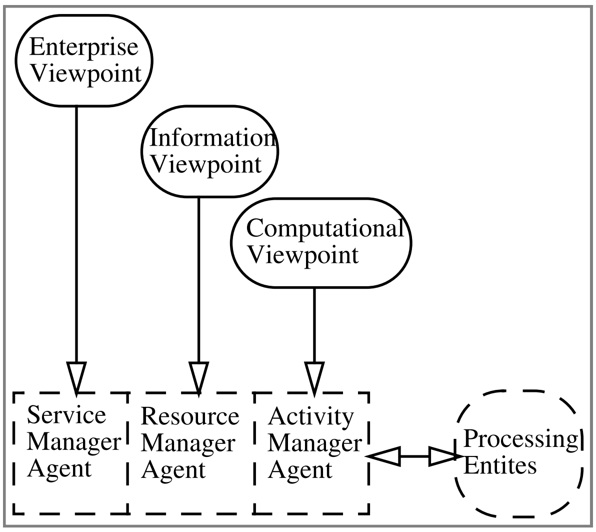


*Figure 1 - Adapting Viewpoints to MAS*

### 3.2. Overall Architecture

Given the set of agent classes we define above, we are going to issue an architectural proposal for structuring open distributed systems according to service and agent concepts. We will consider here the key idea of service as a main guideline in structuration of systems. We consider henceforth that a system is characterized by the set of services it can provide to its environment.

A service can be structured as follows:

- one or many Service Manager Agents responsible of the *policy of the service* encompassing its *use* by the environment and its eventual *collaboration* and *cooperation* with other services. They have the same lifetime than the service,
- one or many Resource Manager Agents responsible of the *local resources* necessary to the service and which it owns and manages. Their lifetime is up to the needs of the responsible Service Manager Agents,
- one or many Activity Manager Agents whose *goals* are determined by some Service Manager Agent. Then, they receive from that Service Manager eventual collaborative/cooperative and cognitive capabilities necessary to fulfill that goal. Their lifetime can end with the definitive success or failure for the assigned goal.

The relevancy of the service notion here is its reflexivity. In complex services, one can undertake decomposition and each Service Manager can be considered - with its attached Resource Managers and Activity Managers - as an elementary service. This decomposition allows to consider services at a granularity level which does not carry too much complexity. So test validation and verification can be achieved in a satisfactory way on elementary services before combining them into more elaborate ones.

The delegation of reactive tasks to Activity Managers by Service Managers can be done one the fly and when needed. The Activity Managers do not need in that case any more collaborative or cognitive aspects. They are only created to perform a given task on behalf on an Service Manager. The Activity Managers can then be validated using formal methods in their reactive aspects. This possibility is left up to system designer and must be evaluated in front of the level of validation one can need in a given application domain.

An obvious advantage of this proposal is the fact that one can isolate the information managed (resources) in the Resource Manager Agents which can be validated formally. We therefore make sure that whatever the collaborative/cooperative and cognitive activities are, the resources will not be corrupted. Resources are critical for a service because their states determine the correctness, safeness and reliability of its behavior. For these reasons, it is important to verify and validate thoroughly their concerns.

## 4. The AOC language

The AOC language is used to specify the reactive aspects of Resource and Activity Managers Agents. It enables to describe a specification model and a verification model [GER 97a] [GER 97b]. The specification model enables 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.

### 4.1. The Specification 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 their own goals, agents have triggers that allow them to run autonomous behavior along with their interactions. Triggers are very important for the achievement of agent-oriented behaviors because they extend the request/reply based interactions between agents. To control the interactions and avoid fault propagation, an agent can have exceptions. Exceptions are executed whenever the agent considers information coming from its environment (mainly results of operations) to not match specified requirements. Then the agent runs the exceptions which allow it to protect itself from what is considered to be erroneous behavior or results.

As we deal with the validation of the organization of interacting agents, we focus on the formalization of the interactions between the agents and on the reactive capabilities supporting them. So we identify some interaction patterns that can be formally proved. These are based on a classification of interactions and their structuring according to some rules. The classification we propose encompasses two groups of interactions built upon the RM-ODP interactions. The *point-to-point interactions* take place between two agents and act upon their current states without modifying their goals and plans. 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.

This classification of interactions enable the identification of patterns of high level interactions. For this, two levels of structuring 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 that allows the agent exporting the service to enforce the other agents to avoid violating integrity constraints on the local resources.

At the second level of structuring, a high-level interaction is defined as *a set of services attached to quality of service expectations and sequencing rules according to the achievement of that quality.*

This notion of interactions enables plans to be built that involve many agents and take into account their dependence on their environment. Interactions allow organizations to be built 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 with which the agent with a desire must interact in order to satisfy it. Furthermore, interactions allow new or elaborate services to be built by combining preexisting validated and verified services or agents. This enhances reduced time to market when building new services.

### 4.2. 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 of the correctness of a specification. 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 allows a statement of 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 about 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.

## 5. Conclusion

The objective of the ODAC project is to define methods and tools based on a formal approach so that a designer of telecommunications services can specify and implement a new service in the form of a Multi-Agent System (MAS). Our goal is to supplement the current approaches of the telecommunications industry by the use of the agent paradigm enhanced with formal methods. So we have defined a service-based architecture of an agent-based system by adapting the ODP viewpoints to the agent paradigm. The benefit is to separate and organize the many concerns of agent systems and to make their dependencies more manageable. We can then isolate the reactive aspects from the collaborative/cooperative and cognitive ones. These reactive aspects can be specified and verified in a formal way by the use of the AOC language we propose and that is in conformance with the ODP computational viewpoint concepts.

## 6. References


[DIA 96] A. Diagne and P. Estraillier, *Formal Specification and Design of Distributed Systems*, in Proc. of the 1st IFIP Workshop on Formal Methods for Open Object-based Distributed Systems (FMOODS'96), Chapman & Hall (Ed), Paris, France, March 1996, pp341-356

[DIA 97] A. Diagne, *Architectural Concepts for Agent Paradigm: A Way to Separate Concerns in Open Distributed Systems*, in Proc. of the 2ndIFIP Workshop on Formal Methods for Open Object-based Distributed Systems (FMOODS'97), Chapman & Hall (Ed), Canterbury, UK, July 1997, pp387-398

[GER 97a] M.P. Gervais and N. Ruffel, *Design of telecommunications Services Based on Software Agent Technology and Formal Methods*, in Proc. of the IEEE Globecom'97, Phoenix, USA, November 1997, vol. III, pp1724-1728

[GER 97b] M.P. Gervais and A. Diagne, *Enhancing telecommunications Service Engineering with Mobile Agent Technology and Formal Methods,* IEEE Communications Magazine, July 1998

[HUR 95] W.L. Hursch and C.V. Lopes, *Separation of Concerns*, Tech. Rep. NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, USA, Feb. 1995

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

[MKA 96] D.P. McKay, J. Pastor, R. McEntire and T. Finin, *An Architecture for Information Agents*, In Advanced Planning Technology, AAAI Press, Menlo Park, CA, USA, May 1996

[PIT 95] J. Pitt, M. Anderton and J. Cunningham, *Normalized Interactions Between Autonomous Agents : A Case Study in Inter-Organizational Project Management*, In Proc. of COOP'95, Antibes-Juan-Les-Pins, France, Jan. 1995

[SCORE] RACE Project n°2017, *The SCORE Service Creation Model*, Deliv. D105-I, Dec. 1994

[THI 95] C. Thirunavukkarasu, T. Finin and J. Mayfield, *Secret Agents - A Security Architecture for KQML*, In Proc. of ACM/CKIM'95, Agent Workshop, Baltimore USA, Dec. 1995

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

[WOO 95] M. Wooldridge, J. Muller and M. Tambe, *Agent Theories, Architectures and Languages : A Bibliography*, in Proc. of the Intelligent Agents II, IJCAI-95 Workshop on Agent Theories, Architectures and Languages (ATAL'95), LNAI n°1037, Springer Verlag (Ed), Montreal, Canada, August 1995


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