Design of Telecommunication Services Based on Software Agent Technology and Formal Methods

Marie-Pierre GERVAIS(*) and Nicolas RUFFEL
Université René Descartes(*) - Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6 (LIP6)
Tour 65-66, bureau 209
4, place Jussieu - 75252 Paris Cedex 05 - France
tel : + 33 1 44 27 73 65 - fax : + 33 1 44 27 62 86
{Marie-Pierre.Gervais, Nicolas.Ruffel}@lip6.fr

This paper will appear in the Proceedings of the IEEE Globecom'97 Conference

Abstract

We propose in this paper an approach based on the software agent technology and the formal methods for designing telecommunication services. This approach aims to identify agent and interaction-oriented design patterns and to describe them with a formalism enabling to prove their quality. These patterns are based on a classification of agents interactions using the interaction model of the Reference Model of Open Distributed Processing (RM-ODP). The formalism offers a methodology for validation and verification based on Petri nets. With such an approach, it is possible to specify some components with proved properties that guarantee their quality. We illustrate it with an example: a Multi-Agent System of a contract monitoring.

1. Introduction

Thanks to the technology evolution, an increasing number of telecommunication services is available to the users for any kind of media. Some of them improve the basic call process such as freephone or callforwarding, others expand the Internet services while others offer mobility supporting the voice transport (GSM, D-AMPS or PDC) as well as data transport (pagers or nomadic computers).

However, this electronic market is only at its infancy and is going to grow up. The continuous technology improvements, e.g., introduction of ATM, are not the only reason. Competition among telecommunication service providers, operators as well as manufacturers, especially because of deregulation is also an important factor of the services development. The service providers have to face the challenge of producing services for attracting customers while reducing the development time and the costs. Such a production can be achieved by adopting the software components reuse approach [SCORE]. But making this approach really efficient requires flexibility and adaptability of the components. Moreover interoperability between components must be considered: a service could be the result of a combination of several components provided by different providers. Finally, the quality of the designed service must be ensured, and for this, validation and verification activities have to be performed. In order to be rigorous, these must be based on formal methods.

These requirements for producing telecommunication services need to adopt new solutions. Then the software agent technology is an attractive solution. With this approach, it is possible to specify a telecommunication service as a set of cooperating agents, i.e., a Multi-Agent System (MAS). Agents offer the characteristics of flexibility, adaptability and interoperability required by the combination activity.

But the agent technology is not based on a formal approach. This limit must be removed in order to provide the designer of the MAS with methods and tools based on formal methods that enable to ensure the quality, i.e., the robusteness and the reliability of the MAS.

We propose in this paper an approach coupling the software agent technology and the formal methods for the design of telecommunication services. This approach aims to identify agent and interaction-oriented design patterns and to specify them with a formalism enabling to prove their quality. These patterns are based on a classification of agents interactions using the interaction model of the Reference Model of Open Distributed Processing (RM-ODP). The formalism offers a methodology for validation and verification based on Petri nets. With such an approach, it is possible to specify some components with proved properties that guarantee their quality.

The paper is organized as follows. In Section 2, we draw the scope of the project developed at the LIP6 and which this work is related to. Section 3 describes our interaction model. Section 4 presents the way we validate the interaction mechanisms and the formalism we use for this. Section 5 develops an application of this approach: a MAS of a contract monitoring. Section 6 concludes this paper.

This work is supported in part by grant 1B 95 from CNET (French Research Center of Telecommunication)

2. Scope and Requirements

The LIP6, in association with CNET, started a project in 1996 for the telecommunication services creation. The aim is to provide a telecommunication services designer with methods and tools in order he/she can formally specify a service as a set of cooperating software agents, i.e., a Multi-Agent System (MAS). The agent concept we deal with is a computer-oriented view of an everyday life agent (such as a travel, estate or insurance agent) . So an agent is a software entity with the characteristics of delegation, customization, contract monitoring and services combination.

As illustrated in Figure 1, 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 [RM-ODP]. RM-ODP is an architectural framework for the design and 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 architecural 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 [Stefani95]. The five viewpoints are the entreprise, information, computational, engineering and technology viewpoints. The *entreprise 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. The *engineering viewpoint* focuses on the structures and the components (the infrastructure) necessary to support distributable objects. The *technology viewpoint* focuses on implementation choices and the realization of the system. The RM-ODP prescribes for each viewpoint a set of rules, called *viewpoint language*, each complying system must obey.

We focus here on the computational viewpoint of our approach. For the other viewpoints, especially the engineering one, interested readers can look at [Merlat97].

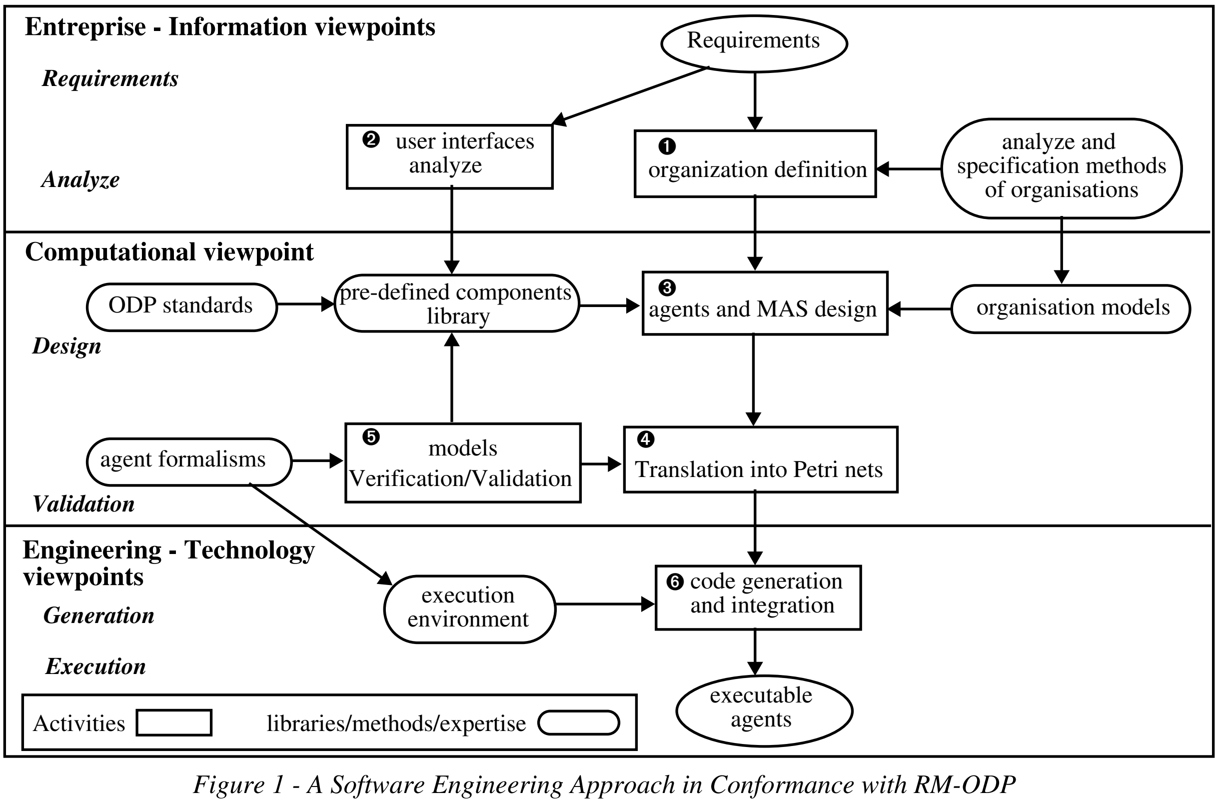

Figure 1 - A Software Engineering Approach in Conformance with RM-ODP

The computational viewpoint enables to specifiy a telecommunication service by means of a computational language. This must enable the specification of a components combination resulting from the derivation and the refinement of agent and interaction-oriented design patterns. This combination requires the identification of agents that must be selected in a library of pre-defined agents. This 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. Then the language must support an interaction model. At a fine grain level, it must enable to define new agents and then to support an agent model. Finally, 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 verification techniques in addition of simulation.

We are currently developing such a computational language. It is based on two complementary languages already defined at LIP6, that are BRIC and OF-Class languages [Ferber95][Estraillier96]. BRIC enables to describe basic modules and to build an agent by composing these modules. It is well-suited to describe an agent model. OF-Class focuses on the external view of an agent as seen by a designer. It is well-suited to the description of an interaction model.

The reminder of this paper describes the interaction aspect of the language, i.e., its coarse grain level. We first present the interaction model we have defined. Then we explain how we identify interaction patterns from this model and how we formalize and validate them by using the OF-Class language. An example is then developed to illustrate this proposal.

3. The Interaction Model

A computational specification is the description of a telecommunication service as a combination of interacting agents. As we deal with the validation of this combination, we focus on the formalization of the agents interactions. So we identify some interactions patterns, i.e., interactions mechanisms that can be formally proved. These patterns are based on a classification of interactions we propose and that is built upon the low-level mechanisms described in the RM-ODP.

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 intermediate states of the agent, i.e., they do not modify its behavior. The 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 [Chess95].
- The *multi-point interactions* take place between several agents. They modify the behavior 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 [Rose94].

4. Specification and Validation of Interactions

We use the above classification to identify some interaction patterns. For this, we consider the ODP interactions as basic ones from which high-level interactions can be built. A high-level interaction is specified as a set of ODP interactions[1], organized according to a usage pattern describing the invocations sequence, the methods parameters and so on. This enables to use a high-level interaction in the same way as a basic one. Only the number and the complexity of parameters (i.e., arguments) are higher than those of a basic interaction. Then the user does not know *a priori* which basic interactions are performed during the high-level interaction execution.

This mechanism is modelled by using the OF-Class language [Diagne96]. This implements the ODP concepts of the computational viewpoint. It is object-based as it enables to create components through instantiation. A component is an entity that manages resources and provides processing for the access and the modification of the resources. Each component provides services that are exported through interfaces as a set of operations. In addition to this ODP concept of operational interface, constraints on the invocations sequence are expressed as a usage pattern of an interface. Any component that 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). A component can also provide non-operational interfaces, i.e., signal and flows interactions.

An OF-Class specification is an entry point towards a validation and verification environment based on colored Petri nets (Figure 2). An interactions validation is carried out on the original specification (❶). Then this is translated into a modular colored Petri net model, called the OF-CPN model (❷). The transformation is fully automated and supported by a tool integrated in the AMI environment, which is a framework dedicated to formalization of software development along the life-cycle [MARS94]. The translator tool guarantees the consistency between the two models. Other validation and verification activities can then be achieved from the OF-CPN model (❸). They include the validation of invariants and of accessibility (liveness) and the search of blocking or divergent states. A semi-formal verification, that is simulation, is also provided. All these activities are supported by tools of the AMI environment.

1. These can be implemented as messages exchanges or RPC

.

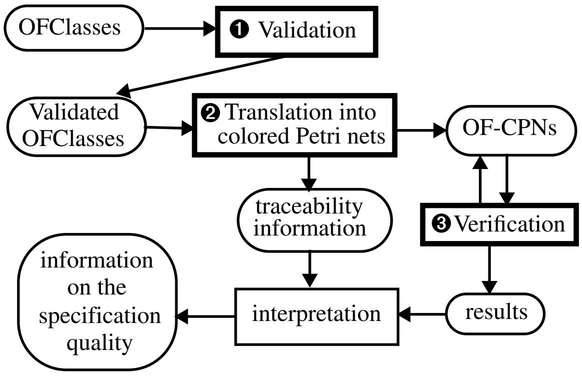

Figure 2 - The Validation and Verification Methodology

The concept of a usage pattern in the OF-Class language enables to organize the basic interactions. This allows to directly validate the high-level interactions. Actually these are specified by means of the OF-Class language based on the basic interactions description. They can be directly used as interaction patterns providing pre-defined components with proved properties.

5. An Application

The above approach has been applied to an example: a MAS of a contract monitoring [Ruffel97].

5.1. The context

The MAS for the contract monitoring takes place in the context of the electronic market of services. This market is composed of two kinds of servers (Figure 3):

- a *secondary server* provides services through a network (e.g., servers of airplane or railway companies) ;
- a *primary server* is an «agency» server that acts as intermediary between a customer and the secondary servers. It provides a customer with the management of the services use (e.g., a travel agency).

The primary server must fulfil the requirements of its customer in terms of Quality of Service (QoS). This represents the satisfaction criteria of the customer such as the service availability, its reliability, its cost and so on. These are expressed through a contract between the customer and the primary server, namely the *Customer_Contract*. This contract is depending on the QoS of the secondary servers. Actually, in order to provide its Customer_Contract, the primary server uses services of secondary servers and therefore it concludes agreements with these secondary servers. Then the primary server guarantees the QoS of the Customer_Contract based on the QoS the secondary servers guarantee it through the subcontracts.

The result is a distributed production chain. Each link of the chain, i.e., the secondary servers, aims to fulfil the contract it supports. The links are independent from each others and they cooperate only to fulfil their subcontract.

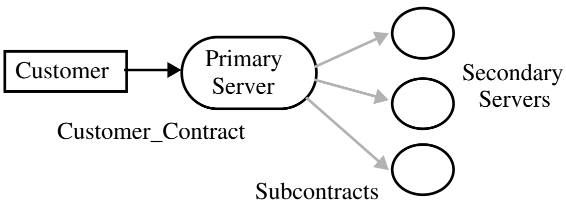

Figure 3 - Context of the Electronic Market

5.2. The Contract Monitoring

The aim of the contract monitoring MAS is to supervise the contract progress and in any case of problem, to find the best substitution solution. As the production chain is distributed, we choose to distribute the monitoring by implementing a μ-agent in each secondary server. Its task is to supervise the local subcontract. A coordinator agent is implemented on the primary server to manage the Customer_Contract. Each μ-agent is waiting for events related to the subcontract it is monitoring. These events come from its server, or they are routed by the server but come from other μ-agents. If an event calls the guarantee of the subcontract QoS into question, i.e., some constraints are no more fulfilled, then the μ-agent tries to find locally the best solution of substitution. However, a modification in a subcontract may influence the others because of side effects. To avoid this, a total ordering on the subcontracts is defined according to priorities: a μ-agent on a secondary server is not allowed to change its subcontract if this modification affects some subcontracts with higher priorities. If this affects subcontracts with lower priorities, then the μ-agent must inform the corresponding μ-agents in order that they find a new solution too. If a local solution cannot be found by a μ-agent, then it informs the coordinator agent and this initiates a global search of a new solution. In order to preserve the causal ordering of the messages received by the coordinator agent, an algorithm of management of logical clock on messages is used [Mattern88].

5.3. Specification and Validation of the MAS

The MAS is composed of the μ-agents and the coordinator agent. Its environment is the set of the servers (primary and secondary). We first have identified the interactions occurring in the system. On one hand, there are some interactions between a μ-agent and its server (resp. the server and the μ-agent). These are point-to-point interactions. On the other hand, there are some negociations between the μ-agents when a new solution must be found. These are realized through the arbitration of the coordinator agent.

Once the interactions are determined, they can be expressed in the OF-Class language as services. The Figure 4 illustrates the graphical representation of the MAS. Three classes of components are identified, namely the primary server (*Primary*), the secondary server (*Secondary*) and the μ-agent (*AGT*). Each component exports or imports services. For example, a μ-agent interacts with its secondary server to supervise its subcontract, i.e., it subscribes to some events (importation of the *Subscription* service) and it collects notifications on these events (importation of the *AGT_Announce* service). It can also look for a new local solution and for this, it imports the *BD_req.* service. A μ-agent must be notified by the primary server that it must locally modifiy its subcontract (exportation of *Contract_modif.* service) or that a new global solution must be found (exportation of *Global_comput.* service). Before looking for a new global solution, the μ-agents must be synchronized by the primary server (exportation of the *Synchro.* service). The primary server can also kill a μ-agent (exportation of the *Self-destruction* service). Finally it can send information to the secondary server (exportation of the *PRI_Announce* service).

The set of these services constitutes an interaction pattern for any kind of application related to a contract monitoring.

Once we get this specification, it is automatically transformed into OF-CPNs. These are not represented because of their size, but some information on these models are provided in Table 1.

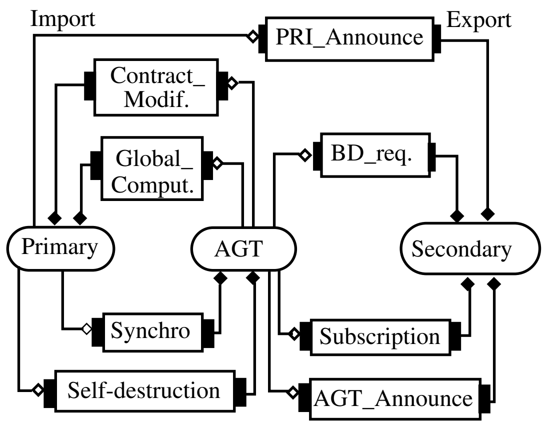

Figure 4 - Graphical OF-Class Representation of the MAS

Table 1 : OF-CPNs

	places	transitions	arcs
AGT	68	44	153
Secondary	45	28	114
Primary	47	31	141

The analysis of these OF-CPNs by means of AMI tools enables to prove the following control properties.

The system is never blocked: when a μ-agent reaches a state in which it cannot guarantee the QoS of its subcontract, it can always find a new local solution or ask to the primary server to initiate the global search of a new solution. When there is no processing in the system, this is in a stable state, i.e., each subcontract is fulfilled and the QoS is guaranteed. Finally,

no global computation can be done while a synchronization is ongoing.
This analysis of the control properties of the system enables to determine that it is safe. This MAS of contract monitoring then constitutes an interaction pattern that can be reused by any application which needs such a service, e.g., an electronic travel agency, or an estate agency or anything else.

6. Conclusion

We have proposed in this paper an approach based on the software agent technology and formal methods for designing telecommunication services in a formal way. It provides a telecommunication service designer with a language that enables him/her to specify a new service as a set of interacting agents. For this, we propose a classification of interactions based on the interaction model of RM-ODP and from which we identify some interactions patterns. These are formally proved by using the OF-Class language. The exemple of the MAS for a contract monitoring application illustrates the approach and the specification and validation tools. We identify an interaction pattern for this kind of application and we prove a set of control properties. This pattern can then be reused in any context in which a contract monitoring is needed.

7. References

[Chess95] D. Chess et al., *Itinerant Agents for Mobile Computing*, IEEE Personal Communications, Vol. 2, N°5, Oct. 1995, pp34-49

[Diagne96] A. Diagne and P. Estraillier, *Formal Specification and Design of Distributed Systems,* In Proc. of the 1st IFIP Workshop FMOODS'96, Paris, March 1996.

[Estraillier96] P. Estraillier and F. Kordon, *Structuration of large scale Petri nets : an association with high level formalisms for the design of multi-agents systems*, in Proc. of IEEE International Conference on Systems, Man and Cybernetics, Beijing, China, Oct. 1996.

[Ferber95] J. Ferber, *Basis for Cooperation in Multi-Agent Systems*, In Proc. of the 95 European Conference on Cognitive Science, St Malo, France, 1995.

[MARS94] MARS Team *The CPN-AMI Environment version 1.3*, Université Pierre & Marie Curie, http://www-masi.ibp.fr/~framekit

[Merlat97] W. Merlat et al., *Mobile Agents for Dynamic Organizations: the Conversational-Agent Paradigm*, to be published in Proc. of the European Workshop MAAMAW'97, Sweden, May 1997.

[RM-ODP] ISO/IEC IS 10746-x — ITU-T Rec. X90x, *ODP Reference Model Part x,* 1995.

[Rose94] J. Rosenschein and G. Zlotkin, *Designing Conventions for Automated Negociation*, AI Magazine, 1994.

[Ruffel97] N. Ruffel and M.P. Gervais, *Un modèle d'interactions orienté-agent pour la conception de services de télécommunication*, Poster Session in 5th JFIADSMA'97, La Colle-sur-Loup, France, April 1997

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

[Stefani95] J.B. Stefani, *Open Distributed Processing: an Architectural Basis for Information Networks*, Computer Communications, Vol.18, N°11, Nov. 1995, pp849-862