This paper will be published in the 6th International Conference on Intelligence in Networks ICIN'2000

SPECIFYING AND VERIFYING THE BEHAVIOR OF TELECOMMUNICATIONS SERVICES

Jean-Fran ois DAUCHEZ(*) and Marie-Pierre GERVAIS(**)
{Jean-Fran ois.Dauchez, Marie-Pierre.Gervais}@lip6.fr
Laboratoire d Informatique de Paris 6 (LIP6)
Universit Paris 6(*) - IUT Paris 5(**)
UPMC - LIP6
P le : M.S.I. SRC
8, rue du Capitaine Scott - F75015 Paris

Abstract

This paper presents the work currently achieved in the ODAC[1] project. This proposes to promote the agent technology by defining 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-Agents System. It aims at providing a methodology for specifying and verifying the behavior of telecommunications services based on the Reference Model of Open Distributed Processing developed by the International Standardization Organization (ISO) and the International Telecommunication Union Telecommunication Standardization Sector (ITU-T).

Keywords: services creation, multi-agents systems specification, validation and verification

1. Introduction

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 to define an architecture and a methodological framework with its support [1]. A major problem that designers have to face is a consistency problem. There is no easy way for a designer to define a new service in a system because of compatibility problems between the existing services and the new one. On the other hand, various works aim to define systems based on the agent technology and show encouraging results [2]. Indeed, it is easier to define a service with a greater abstraction level. However, these systems are still less used because of the maintenance problem or because of the difficulties to debug such applications [8].

To help a telecommunications service designer in his/her task, we propose to consider a telecommunications service[2] as a community of agents. As an agent can be itself a community, the problem is then the control of the behavior of all the system components.

The ODAC project proposes to promote the agent technology to specify telecommunications services [3]. It provides a specification methodology based on the Reference Model of Open Distributed Processing (RM- ODP) that allows a clear separation between the many concerns of the system. To help this specification methodology, there are tools provided to verify a specification and to extract some qualitative properties.

This paper describes the current work achieved in the project. First, we define the various agent characteristics considered. Then, we describe the specification methodology that we are proposing. Finally, we present validation and verification activities that can be realized on the specification.

2. The Agent Characteristics

The agent world is a wide study area and it is difficult to enumerate all the different definitions of the word ˙agent¨. We will stick to the distributed artificial intelligence definition: in our view, an agent is a software entity that is autonomous, social, reactive, pro-active and sometimes mobile [5]. A multi-agents system is a community of agents working together to meet an objective [4]. Each agent is responsible for a part of the system objective, in other words, the agent objectives are to reach a part of the global objective. We consider the agent objectives as the smallest possible part of the system objective. The main reason of the agent existence is that it must be able to achieve at least one objective.

1 ODAC stands for Open Distributed Applications Construction
2 In the following, the telecommunication service to be designed is identified by the term «system»

An agent is autonomous in the sense that it has the total control of its execution to achieve the objectives. Thus it is able to control the schedule of its tasks. However, this does not mean it is able to compute all the needed tasks to fulfill the objectives. It might also use services provided by other agents.

An agent is social as it must be able to use and to give back services from or to other agents. We consider here two different ways to communicate. The first one takes place in a relation client-server. An agent that needs to evaluate a process and which cannot do it by itself must ask for it to another agent. This request can be either some interrogations, i.e., there is a need for an answer, or some announcements, i.e., no answer is needed. The second way to communicate is the stream. In a distributed system, the information can be held by different parts of the system. A stream is a way to distribute to the other participants the information they need.

An agent is reactive because of its responding capabilities to the environment changes. These can be related for example to its dependence on other agent services. In an open system, the existence of the services offered by an agent is not continuous in time. So, in that case, an agent can loose a service needed for its objectives or can find a better service to achieve them with a better quality.

An agent is pro-active means that an agent is able to achieve its objectives whatever events, reactions or interactions occur in the system. It has a guideline that enables it to achieve its objectives by the execution of different tasks. Thus it is able to refuse some interactions or reactions to complete its objectives. This can be seen as the fairness between different tasks, some related to the provision of services while others related to the objectives accomplishment.

An agent is mobile when it can move in the system to be close to the resource it needs. The agent mobility, which is sometimes a controversial point, can be justified by performance improvements, rights increasing or by changes made to its environment. Movement implies the existence of an adequate environment or the encapsulation of the agent in a shell.

3. The ODAC Specification Methodology

Specifying telecommunications services by using multi- agents system introduces problems of maintenance difficulties such as modifications of the specification. A project manager who conducts the service design activities needs a structured set of guidelines, a homogeneous terminology, concepts with an efficient abstraction power and a comprehensive and incremental history of the project. The ODP reference model fits these requirements by proposing a panel of different viewpoints to guide the designers [4]. The concepts are well defined and they provide a level of abstraction that permit to deal with the complexity of the design process.

The ODAC specification methodology is based on three ODP viewpoints, namely the enterprise, information and computational viewpoints. The specification of a new service consists in describing this service according to the viewpoint concepts [3]. It should be noted that although in ODP there is no sequence between viewpoints, the methodology recommends to start with the enterprise viewpoint as it lets the designer to express the needs of his/her system. We use it as the requirement engineering and the analyses of the system. We introduce the computational viewpoint as the service design. The information viewpoint is used to describe the information in the system as we will detail it later.

3.1. Enterprise Specification

The enterprise specification describes the systems in terms of objectives, roles, policies, resources and communities [6].

Firstly, the service designer must determine the service objective and then how to reach this objective. This process can lead to the need of refining this objective several times. The refinement stops when elementary objectives are identified and the decomposition is no more possible.

The designer assigns a role to each elementary objective. This role is then responsible to fulfill this objective.

A role implies some actions. For example, in a virtual travel agency, one of the roles is to keep up to date the file of a client after a transaction. Actually, a flight can be cancelled and then another travel has to be scheduled. So, to keep it up to date means accepting a flight cancellation message and asking to reschedule the travel. The job is finished when the departure announcement is received. Here, three actions are defined for the role: receive a cancellation, ask to compute a new travel and receive the announcement departure.

Organizing the role actions determines the behavior of this role.

The policies are the set of role interdictions, permissions and obligations that a role respects in the relation with other roles. They are part of a contract. In our example, the role can impose to a client (i.e., another role) the ticket exchange.

Resources do not have behavior but they have a lifecycle. The enterprise viewpoint specifies which roles are responsible for the creation, the use and the deletion of the resources in the communities.

A set of roles and resources constitutes a community. The communities have the same characteristics as the roles, plus a resource policy. The resource policy describes the lifecycle of the community resources. The actions of a community are the or a subset of the actions of roles it contains. The behavior of the community, called the activity, is a graph of the community actions where an action is made possible by the occurrence of all immediately preceeding actions. The community contract is the union of the roles contracts and eventually some new interdictions, permissions or obligations.

In the example of a virtual travel agency, the roles defined, namely the travel agency, the airplane company and the client constitutes a community. They all want to accomplish the same objective: the departure of the client.

We still did not explain the place of an agent in this viewpoint. As we said, an agent is autonomous. So it is the entity that encapsulates the smallest objective of a role. It can encapsulate more than one. A community for example can be encapsulated in an agent. An aspect of the interaction is shown in the community behavior. The fact that an agent is intentional comes partially from the roles or the communities behavior. Its intention is to realize the different actions as they are described in the behavior.

3.2. Information Specification

The information specification describes the information semantics and the processing information semantics in a system. The description is divided in three parts: invariant schema, static schema and dynamic schema [6].
The invariant schema describes the boundary of the different types. A type is a set of finite values and can be composed of other types. This schema enables the limitation of the exchangeable values in the system. We use the same kind of types as those defined in the Interface Definition Language (IDL).
The static schema defines the kind of states the agent can take and more especially the initial, the final and eventually, some intermediate values.

The dynamic schema describes the actions that will modify the information. These actions linked together permit to reach, starting from the initial value described in the static schema, the other values. An action has to be seen here as an elementary action.

There is a correspondence between actions described here and those defined previously in the enterprise viewpoint. They are taken from the roles by defining states they will reach. The information specification describes essentially the pro-activity characteristic of the agent. It also describes the evolution of the agent's knowledge.

3.3. Computational Specification

A computational specification describes the functional decomposition of the system, in distribution transparent terms, as [6]:

- a configuration of agents;
- the internal actions of those agents;
- the interactions that occur among those agents and that belongs to exported or imported interfaces;
- environnement contracts represented in the agents interfaces. They are the set of actions (eventually restricted by parameters types for example) exported conditionally to find the associated imported interfaces.

As seen previously, agents interact through interrogations, announcements and streams. As ODP we put together interrogation and announcement into a same pattern: the operation.

Set of actions of the information specification have a correspondence with operations. An interrogation expresses the need for a result and is composed of an invocation followed by a termination. So, it is seen as a synchronous action while an announcement can be seen as an asynchronous action. The parameters and, for the interrogation, the return values types are defined in the information specification. The operations are put together into interfaces that show what services an agent can offer. Each interface can require services from other interfaces to be usable.

Produced streams are described with a type of data and are put together in stream interfaces.

An agent is reactive. This viewpoint allows the identifications of the events that the agents need to consider. They are described in signal interfaces: emitted or accepted signal interfaces. When an agent must respond to a signal, it can require other interfaces.

An agent is also social. In this viewpoint, it identifies the interactions between the different roles, the protocols they will use.

4. Verification and validation

The specification methodology presented above is based on the separation of concerns reflected in the viewpoint concept. This separation simplifies the verification tasks. Actually, it enables the designer to verify locally each specification or part of it rather than verify the global specification. The verification is based on the system behavior analyze, that is the behavior of the agents set. It consists in verifying the properties expressed by the designer and in validating a part of the entire system [7].

4.1. Verification

The verification means to check if the system will really do what it is expected to do. The verification is applied to each viewpoint specification and particularly to each part of it. It is concerned to verify some properties such as the violation of a behavior, the reachable state, the reachable objective and pre- and post-conditions verification.

4.1.1. Violation of a behavior

As soon as the enterprise specification is available, roles behaviors are described in terms of actions. Moreover, the constraints are defined on the roles in terms of interdictions, permission and obligations. It is then possible to check that the set of constraints does not lead to deadlocks in the role behavior.

To realize this verification, behaviors and contracts described in the enterprise specification are automatically transformed into Petri nets. An action becomes a transition. The pre-conditions of a transition are the post-conditions of the action that permits it and the ones that preceed it. A post-condition is generated each time an action permits another one or each time an action has a successor. An activity represents a connection between two or more Petri nets. In that case, we divide the Petri nets into independent parts and study each of these independent parts as a new behavior.

With these Petri nets, if one permission is missing, then their liveness is not ensured. We must verify that a transition can be fired after an obligation and also that a transition cannot be fired after an interdiction.

We verify the behavior of the agent by checking if all transitions are reachable from the initial state. The set of these verification tasks is achieved by using the CPN- AMI2[3] environment, that is a framework dedicated to

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

formalization of software development during the lifecycle.

4.1.2. The state reachability

The information specification describes states of the system and state changes in terms of actions. It must be checked that each of these states can be reached. This means that for a given state by using the actions enabling the state changes, at least one way to reach this state must be founded. This way is a permutation of a set of actions.

This allows the designer to verify that all the needed actions have been described. It enforces the specification by removing the unused actions.

4.1.3. Objective reachability

Verifying that the objectives could be reached means ensuring that the actions can be done entirely. Helped by the computational viewpoint description, we extract some Petri nets that represent the actions. We have here one action for one Petri net. So, a Petri net is of a lower level than the one mentioned in the Section ˙Violation of a Behavior¨. To avoid a combinatory explosion of the set of system states, instead of combining all these Petri nets, we replace the place that links them by a transition and a place. This transition simulates the value of the replaced net. Then the verification can be done more easily by abstracting a behavior with the corresponding action. Finally we verify that each action can end with a final state that represents the post-conditions of the action.

4.1.4. Pre- and post-conditions verification

Pre- and post-conditions verification is realized by using the behavior described in the computational specification. It determines if each invocation is done with the right types (the pre-condition) and for each operation if the data state will remain in their boundaries (the post-condition).

4.2. Validation

Validation means that the system will do what it is expected to and nothing more. The validation is applied to each viewpoints specification. It is concerned to check some properties such as the identification of new services, the boundaries of types and the identification of new behaviors.

4.2.1. Identification of new services

A T-invariant determines an invariant associated with the transition. We want here to identify the T-invariants of the system modeled as a Petri net built as mentioned in Section 4.1. A T-invariant represents a behavior or a part of it, so the designer must be able to identify the semantics of these T-invariants and check if these T- invariants are or not undesirable new services.

4.2.2. Boundaries of types

In Section 4.1, verification of the pre- and post- conditions aimed to verify the correction of the pre- conditions of an action. Here, we check if all the values defined in the pre-conditions are used. Unused values are removed from a pre-condition by a looping process until no more values can be found. This does not permit to find the smallest boundaries of the types but it determines a reduction of the types. Reducing the numbers of values used in the system can enable the designer to identify a design mistake in the semantics of the concerned type.

4.2.3. Identification of new behaviors

We do not use here the same technique as we did in the identification of new services. We check that the links between the different actions in the information specification and the behaviors in the enterprise specification have a correspondence in the computational specification. They correspond to emission or reception of signals, writing or reading of information in streams or invocation of operations. We check also that no new link or behavior appears. This means that no agent can emit or accept a signal (resp. write or read a stream or invoke an operation) that has never been described before.

5. Conclusion

The objective of the ODAC project is to define methods and tools based on a formal approach so that a designer of telecommunication services can specify and implement a new service in the form of a multi-agents system. Our goal is to complete the current approaches of the telecommunications industry with the agent paradigm enhanced with formal methods. The benefit is to separate the responsibility during the design process and to verify the different properties expressed in each viewpoint. The major contribution of ODAC is the proposal for a specification methodology conform with the ODP standards that supports the formal verification. The methodology is supported by an operational environment currently developed. This will provide design tools and will be integrated into the CPN-AMI2 environment that provides verification tools.

6. Reference

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

[2] 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

[3] A. Diagne and M.P. Gervais, Building Telecommunications Services as Qualitative Multi-Agent Systems the ODAC Project, in Proceedings of the IEEE Globecom'98, Sydney, Australia, November 1998

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

[5] M. Wooldridge. Agent-based Software Engineering. In IEE Proceedings on Software Engineering, 144(1), pages 26--37, February 1997

[6] ISO/IEC IS 10746-3 - ITU-T Rec. X903, ODP Reference Model Part 3: Architecture, January 1995

[7] M.P. Gervais and A. Diagne, Enhancing Telecommunications Service Engineering with Mobile Agent Techology and Formal Methods, IEEE Communications Magazine, Vol. 36, n° 7, pp38-43, July 1998

[8] M. J. Wooldridge and N. R. Jennings (1998) "Pitfalls of Agent-Oriented Development" Proc 2nd Int. Conf. on Autonomous Agents (Agents-98), Minneapolis, USA, 385-391