This paper will be appeared in the Proceedings of the International Conference on Intelligence in Networks
(ICIN'96 - Bordeaux - Novembre 1996)

Formalization of Service Creation in Intelligent Network

Marie-Pierre GERVAIS(*) and Alioune DIAGNE
Université René Descartes(*) - Université Pierre & Marie Curie
Institut Blaise Pascal - Laboratoire MASI (CNRS UA 818)
4, place Jussieu, 75252 PARIS Cedex 05, FRANCE
{Marie-Pierre.Gervais, Alioune.Diagne}@masi.ibp.fr

Abstract

The service creation in Intelligent Network is a software quality problem that encompasses several activities, namely specification, design, implementation, deployment and maintenance. In order to ensure the software quality, validation and formal verification are required. Validation of a service consists of verifying the global coherence of its specification and design, while verification consists of formal proof of structural and behavioral expected properties.
We propose in this paper a multi-formalism approach, coupling the use of Object-Oriented (OO) and Petri Nets (PN) paradigms, which enables a designer to validate and to verify a service specification. We provide him a proving toolset in order to achieve these activities. This paper describes this proving toolset and its experience on an exemple.

1. Introduction

Intelligent Network has been introduced in order to facilitate the development of new telecommunication services. It is an architectural concept that separates the call control from the service processing. It enables to consider a telecom network as composed of two distinct parts, one dedicated to the transport function and the other one responsible for the service realization. With such an architecture, a predominant role is given to the services aspects. One major question is to know how to enlarge this services market by introducing new services. The difficulty is to specify and to design a service in order to integrate it into an operational system without altering the existing services operation. An example of this difficulty is the well-known problem of the feature interaction risk, which a classification is given in [Cameron94] and some resolving approaches are presented in [Bouma94].
We consider that the service creation is a software quality problem. A telecommunication service is a distributed application built as a set of interacting components. Its creation encompasses several activities, namely specification, design, implementation, deployment and maintenance. In order to ensure the software quality, validation and formal verification are required. Validation of a service consists of verifying the global coherence of its specification and design, while verification consists of formal proof of structural and behavioral expected properties. Validation and verification must be applied to each step of the creation process, given that the later an error is detected, the harder its correction is. This way of service creation prevents to pass on errors from phase to phase in the life-cycle. In order to be rigorous, the service creation needs to be based on formal methods. A number of formal methods have been proposed for that purpose, including finite state-machines, transition systems or variants of temporal logic.
We propose in this paper a multi-formalism approach, coupling the use of Object-Oriented (OO) and Petri Nets (PN) paradigms, which enables a designer to validate and to verify a service specification. This approach and the associated proving toolset are presented in Section 2. Then we develop in Section 3 an illustration of its use by providing an exemple of a service specification and the explanation of the specification analysis. Section 4 provides some concluding remarks.

2. The Multi-Formalism Approach

In the service creation, the phases of specification and design are frequently performed by using Object-Oriented methods (e.g., OMT and Class/relation). Advantages of OO methods include :

- structuring facilities : an application is a distributed software built as a set of elementary interacting components (objects) according to the client/server paradigm : a component (a server) offers services, which can be invoked by the others (the clients),
- seamless suitability to the different stages of the application creation : the application is built by progressive refinement,
- encapsulation: an application is characterized by the interactions it supports rather than by the way it is designed.

Unfortunately, OO methods do not provide validation and verification features to evaluate the correctness and compliance of the model. So we propose to integrate the Petri Nets proving power in an OO methodology. PNs are well-suited to reactive and distributed systems such as telecommunication systems. This formalism is richer than others generally used in the telecommunication area (e.g., SDL or Estelle) since its provides verification techniques in addition of simulation. However, Petri nets, as well as other formal methods, have substantial practical limitations. People require extensive training to use them because the tools that support them are often difficult to incorporate into the standard production software development environment. Moreover, PN lacks of structuring facilities. Coupling the use of PNs with the OO paradigm enables to overcome such limitations. Our claim is to provide design engineers with a proving toolset, provided that they structure their models in a relevant way. The models structure is supplied as design patterns suggested by the service architecture and the properties to be verified.

2.1. The Proving Toolset Description

The proving toolset is composed of two models, namely the OF-Class[1] model and the OF-CPN[2] model [Diagne96a]. The OF-Class model, which is an object-based model supporting the design and part of the validation activities, is the entry point of the proving toolset (Fig. 1). It is formally and automatically associated with a modular Petri net model called OF-CPN model, which supports the remainder of the validation and verification activities. This way of integration allows to hide the CPN to the design engineers.

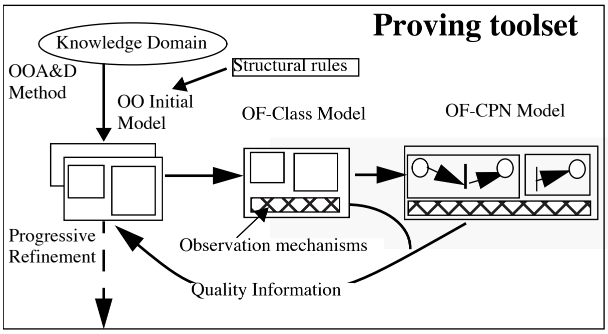

Figure 1: The Two Models of the Proving Toolset

Design patterns, i.e. structural rules, are given in the OO environment. The OO initial model of the application is then designed according to an OOA&D[3] method with respect to these design patterns. They allow to transform this initial model into an OF-Class model. This OF-Class model can be overloaded with observation mechanisms. It supports the activities of validation and verification, which provide results that are interpreted back to the initial model in terms of relevant quality information. The initial model can then be refined according to the supplied information. Validation enables to obtain information related to the global coherence and the completeness of the model. It is a first step to evaluate the correctness of interactions between the application components. Verification deals with safety and reliability properties. Safety properties ensure that faults like deadlocks do not occur in the application. Reliability properties ensure that functionalities of the application are correctly fulfiled.

OF-Class Model

The OF-Class model has two description levels, a Micro-Level and a Macro-Level.

The Micro-Level describes the local resources of a component and their possible transformations. A resource is an entity local to a component that can be accessed only through invocations of services offered by that component. Transformations of resources are done by means of elementary actions. These actions are grouped as sets called operations. An operation is a set of actions performing a given semantical transformation on local resources. An operation can issue requests to the environment. It has input and output parameters, local variables and a return code. A component may also trigger some sets of actions when reaching some meaningful states or when some events occur while interworking with the environment. These mechanisms are slightly different from operations because they cannot be invoked by clients. Such actions are called triggers and exceptions. Triggers bear eventually preconditions, i.e. predicates on the resources values or input parameters specifying the state in which they are executed. Triggers can undertake interactions with the environment.

The Macro-Level in OF-Class describes the structural and dynamic links necessary for interaction of components in the service. Structural links allow to compose discrete components in order to build a more complex one. Dynamic links are:

- the offered services exported by a component. An offered service is a set of operations with contractual constraints like precedence or access semantic (synchronous, rendez-vous, etc.). It is a coherent partial view on the behavior provided to the environment for access to the local resources. The contractual constraints give the usage manual of the service.
- the required services from the environment show a given component the way it must use the services. They are provided by other components of the application.

2.2. The Proving Toolset Use

The OF-Class model is mapped on a modular colored Petri net model (OF-CPN). So once the application is described as a set of OF-Classes, we can undertake transformation into OF-CPN. The transformation is fully automated and supported by a tool integrated in the AMI environment, which is a framework dedicated to

1. OF-Class stands for Object Formalism class
2. OF-CPN stands for Object Formalism Colored Petri Net
3. OOA&D stands for Object-Oriented Analysis and Design

formalization of software development along the life-cycle [MARS94]. The OF-CPN model of the application is used to compute the state/transition graph fully describing the possible behavior of each component. This graph is reduced using observation-equivalence called also CFDD-equivalence in process algebra [Valmari94]. Two components are equivalent with respect to such a relation if they consume and produce the same events on their interfaces. The proving procedure consists of considering for a given component of a system its full state/transition graph interfaced with the reductions of the other components that give an exhaustive abstraction of the environment. We can therefore validate the behavior of the component given that abstraction.

3. Case Study

One of the benefits of formalizing application creation is the ability to detect errors that can occur during the application creation. So we illustrate our methodology by addressing the problem of the introduction of a new telecommunication service in an operational telecommunication system. The chosen example is an Intelligent Network (IN) telecommunication system that provides the Call Forwarding Unconditional (CFU) service and in which a new service is added, namely the Terminating Call Screening (TCS) service. The design is incremental and is composed of three steps. Firstly we model a basic telecommunication system that provides the most elementary service, namely the basic call. Then we modify this telecommunication system model to introduce the Intelligent Network (IN) capability, which is the ability to detect an IN service demand, and we specify a telecommunication system that provides the IN Call Forwarding Unconditional (CFU) service. Finally we make the model richer by introducing the Terminating Call Screening (TCS) service.

3.1. The Basic Telecommunication System

A telecommunication system operation is based on the exchange of messages and signals. The specification of such a system enables to validate these exchanges and to verify that the system behavior is in conformance with the requirements. The specification of the basic telecommunication system we made is based on a simplified scenario of a communication between two users. We assume that the network is composed of one single switch, and two lines. By this way, no routing function is taken into account by the switch. We also assume that the switch does not have any dialing errors to manage. Moreover, we omit the charging aspects. Finally, we adopt a unified process of the call release since it does not deal with the dissymetric aspect of the call release, i.e., the switch processes the release in the same way whether the caller first hangs up or the callee.

The OF-Class model related to such a telecommunication system contains two OF-Classes, namely the OF-Class `Terminal` and the OF-Class `TSwitch`. The user is not modelled since he does not belong to the system. His interaction with the system will be taken into account as an external entity that has not to be validated. The OF-Classes interact through their interfaces in order to perform the system function. The set of possible interactions between a terminal and a switch reflects the communication progress and is expressed through interfaces, which constitute the macro-level description of the OF-Class. Moreover, each of these OF-Classes owns resources. For example, a switch manages a set of lines, and a set of connections that result of the ongoing association of two lines. We have simplified the description by characterizing a line with an identifier that corresponds to the terminal number and a state.

3.2. The CFU Telecommmunication System

Based on the specification of a basic telecommunication system, we model a CFU Telecommunication System, that is a system that provides the Call Forwarding Unconditional service. The Functional Entities we consider are the CCF/SSF, SCF and SDF entities[1]. We only consider the user procedures of the CFU service, namely activation, deactivation and use and we study in depth the processing of a CFU number demand. When a user dials a CFU subscriber number, the call must be redirected to the Forwarded-To number provided by the CFU subscriber if the service has been activated. So no specific procedure is required from the user calling the CFU subscriber. Only the system is involved to detect that such a basic call demand is in fact a CFU processing demand.

The OF-Class model of such a system contains the same OF-Class `Terminal` as the previous model, a new OF-class `SSF` corresponding to the modified OF-Class `TSwitch` enhanced with the ability to detect an IN demand, and two other new OF-Classes, namely the `SCF` and the `OFCallForwarding` OF-Classes[2].

The CCF/SSF detects a CFU demand by the processing of detection points (DPs). A simplified DPs processing is considered: only the DP12 is taken into account, that is the processing associated with the acceptation of a call on the terminating side. Actually, before delivering the call to the called terminal, the CCF/SSF determines if it is a basic call or if it is an IN service demand. For that, we assume that an armed DP12 and the corresponding service key, namely the CFU number demand, are associated with the CFU subscriber number. This is realized by using a table that associates the armed DPs for

1. We do not consider the SRF entity that could be used for the user interaction.
2. In order to simplify, we group together the SCF and SDF entities in a single OF-Class.

a service with the user profile identified by his terminal number. This table `tPointTable` is defined as an SSF resource. So any call with call parameters corresponding to a CFU subscriber number is identified as an IN service demand and is suspended while a request is addressed to the SCF for service processing. The CCF/SSF will continue the basic call processing according to the SCF order related to the end of the service execution. Three types of orders are identified :

- continue with same data : the CCF/SSF can resume the basic call processing with the same data, i.e, the calling line identity (`callingLineID`) and the called line identity (`calledPartyNumber`),
- continue with new data : the service execution provides new data by modifying of the initial ones (e.g. the Forwarded-To number replaces the called number). So the CCF/SSF has to resume the basic call processing by taking into account these new data.
- clear the call : following the service execution, the call must be cleared by the CCF/SSF. This response is provided for example in case of error occurring during the service execution. We also use it to terminate a service demand for which no connection request is required, for example the activation or deactivation of a service. In such cases, once the service is performed, we assume that the call can be cleared.

So the OF-Class `SSF` imports from the OF-Class `SCF` the service `StartUp` that is composed of the operation `InitialDP`. The service invocation mode is synchronous. The OF-Class `SCF` models the entity that is responsible for the processing of the SSF requests. When the SSF invokes the operation `InitialDP`, it provides to the SCF one parameter `initialDPArg` that contains the identifier of the service `serviceKey`, the identity of the calling terminal `callingLineID` and the identity of the called terminal `calledPartyNumber`. The `serviceKey` is a discriminator that enables the SCF to determine which service logic program is concerned by the request. So the SCF transfers the request to this service logic program. The macro-level of the OF-Class `SCF` only contains an operational interface. The imported service of this OF-Class is the service `CallForwarding` composed of three operations, the `CF_Activation`, the `CF_Deactivation` and the `CF_Processing` operations. This service is used by the SCF to invoke the suitable service logic program. Its invocation mode is synchronous. The OF-Class `SCF` exports the service `StartUp`, which is used by the SSF. The `InitialDP` operation is the single operation composing the service. The service invocation mode is synchronous. The micro-level description is very simple: only the operations contents are needed. The SCF manages no resource because it only works on supplied data and does not need to have persistent states. It should be noticed that the model of the SCF entity defined in CS-1 identifies several modules, especially the service logic programs. In our model, we have expressed this characteristic by defining two OF-Classes. One is the OF-Class `SCF` that represents the module that receives an SSF request, analyzes it and invokes the suitable service logic program. The other is the OF-Class `OFCallForwarding` and it represents the service logic itself. As a service logic program is a part of the SCF, the OF-Class `SCF` includes the OF-Class `OFCallForwarding`.

The OF-Class `OFCallForwarding` represents the service logic of the CFU service. Its macro-level description contains the exported service `CallForwarding` with its three operations `CF_Activation`, `CF_Deactivation` and `CF_Processing`, corresponding to the user procedures. The usage manual expresses the dependency relation between the `CF_Activation` and the `CF_Deactivation` operations, as the `CF_Activation` operation must always preceed the `CF_Deactivation` operation. On the other hand, the `CF_Processing` operation can be invoked independently from the two others. It should be noticed that we have introduced the assumption that SCF and SDF are grouped together. As the OF-Class `SCF` models the SDF too, we define one resource at the micro-level. This is the list of the subscribers who have activated the CFU service, `CF_activeList`. Only the `CF_Processing` operation is specified into details, which performs a CFU number demand processing.

3.3. The IN Telecommunication System

Based on the CFU system, we now model an IN Telecommunication System, that is the CFU system in which the Terminating Call Screening service (TCS) is introduced. The TCS service enables a user to forbid some calls to be delivered on his terminal. This means that the subscriber has to provide the calling numbers he wants to forbid, and when an incoming call is presented to the switch from one of these numbers, the switch stops this call. As we did for the CFU service, we focus on the user procedures of the TCS service, especially the processing of a TCS number demand. The TCS Number Demand Processing is very close to the CFU processing, since it is also based on the call presentation to the called terminal. So the CCF/SSF detects the TCS demand through the DP12 processing and invokes the SCF by providing it with the service key. Based on it, the SCF invokes the TCS service logic program that determines if the call must be filtered or not. Regarding the response, the SCF will deliver to the CCF/SSF the order to resume or to clear the call[1].

For introducing this new service, the CFU Telecommunication System requires some

1. Once again, we simplify the procedure by omitting the announcement to the caller that he is not authorized to place his call

modifications. The OF-Class `SSF` has to be modified to be able to detect a TCS service demand and to initiate a corresponding request to the SCF. Obviously the DP12 has to be armed. As in the previous model, we use the SSF resource `tPointTable` to associate an armed DP for a service with the user profile identified by his terminal number. So this table has to be modified by adding the association corresponding to the TCS service. This table represents the list of services for which the DP12 is armed and for which the CCF/SSF has to request to the SCF the execution of the corresponding service logic programs during a given call. If several services have to be invoked in the same call, the table is also used to determine the invocation priority and then the sequence of the requests sent to the SCF.

The OF-Class `SCF` has to take into account the new service because we consider that the same SCF performs the two services. Actually, having one single SCF performing the two services, or two SCFs, each of them dedicated to the execution of one service, has no impact on the activities of validation and verification of the system realized by the formalism. The proving process enables to validate the correctness of the specification in terms of expected functionalities and to verifiy its structural and behavioral properties. So it is independent of the assumptions made in the specification itself. Its role is precisely to evaluate that with these assumptions, the specification is correct. So OF-Class `SCF` will include not only the OF-Class `OFCallForwarding`, but also the OF-Class `OFTerminatingCallScreening`, which models the TCS service logic. In the operational interface of the OF-Class `SCF` is added the imported service `TerminatingCallScreening`, composed of the five operations corresponding to the user procedures, `TCS_Activation`, `TCS_Deactivation`, `TCS_AddNumber`, `TCS_RemoveNumber` and `TCS_Processing`. The usage manual expresses the dependency relation between the `TCS_Activation` and the `TCS_Deactivation` operations, as the `TCS_Activation` operation must always preceed the `TCS_Deactivation` operation. The same precedence relation exists between the `TCS_AddNumber` and the `TCS_RemoveNumber` operations. On the other hand, the `TCS_Processing` operation can be invoked independently from the others. This service is synchronously invoked. At the micro-level, the service keys concerning the TCS service and enabling the SCF to determine which service logic program is concerned by the SSF request are added. The OF-Class `OFTerminatingCallScreening` represents the service logic program of the TCS service. Its macro-level description contains the exported service `OFTerminatingCallScreening` with its five operations. Resources are the list of the subscribers who have activated their TCS service, `TCS_activeList`, and for each subscriber, the list of his denied numbers, `TCS_deniedList`. Only the operation `TCS_Processing` is specified into details.

3.4. The Analysis of the IN Telecommunication System Specification

The analysis of the obtained OF-Class model can be realized from different viewpoints, and provides different types of results. We first present the validation and verification realized with the other model of the proving toolset, namely the OF-CPN model. Then we consider other remarks that can be obtained directly from the OF-Class specification.

The Validation and Verification Using the Proving Toolset

The analysis realized with the proving toolset enables the designer to establish that the specification of the system is correct. Once we have a syntactically correct OF-Class model, it is automatically transformed into OF-CPNs. Each OF-CPN is extended by including the usage manuals of its offered services. We determine the initial configuration of the system as follows. Three users A, B and C are considered. A has subscribed to the CFU and TCS services and has activated them. A forwards his calls to the B's terminal and denies calls from C. B is idle and C initiates a call to A.

We compute the state/transition graph for each OF-CPN according to the inital configuration and an exhaustive abstraction of the environment. This contains the expected results of interactions with the other OF-CPNs as well as a representation of the part of the system that is not explicitely modelled (e.g., the user's behavior). This abstraction enables an OF-CPN to guarantee a correct operation on condition that the environment correctly operates. Then the OF-CPN modelling the SSF is combined with the reductions of the other components together in one net called the *extended SSF*.

Computing the state/transition graph of the *extended SSF* shows that a communication can be established between A and C, violating thereby the TCS functionality (Fig. 2). In fact, the precedence relationship between the invocation of the CFU and TCS services in the same call established by the way we manage the table is not correct. In a correct operation, when a call is addressed to a subscriber that has subscribed to the CFU and the TCS services, if the services are activated and if the caller belongs to the `deniedList` of the subscriber, then the call has not to be forwarded. So the first service to be invoked must be the TCS service, then the CFU service. The invocation priority is determined by the table `tPointTable` and then is depending on the management policy of this resource. This policy is the combination of an updating policy and a consulting policy, both must be consistent with each other to guarantee that no interactions between services will occur. So introducing a service needs to modify the resource`tPointTable` and to build a new one, according to the management policy. Once the table is built, our formalism enables to determine the correctness of the building, by verifying the precedence relationship between the invocations.

We had chosen a very simple resource management policy that is the FIFO management. So the oldest service created in the system, namely the CFU service, is invoked first. The verification of the table building has shown that the management policy is not correct.

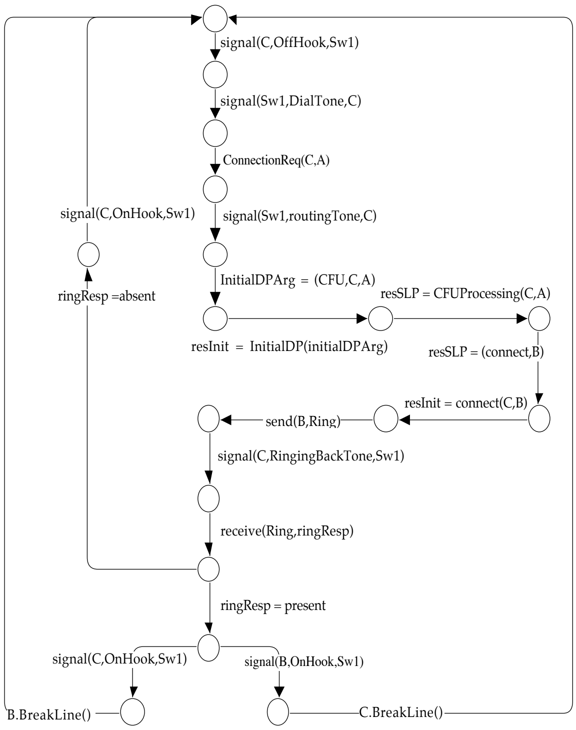

Figure 2: The State/Transition of the Extended SSF

Parallel Invocation of Services: an Alternative Approach

We have applied the IN-CS1 principles, namely single-ended and single point of control services. The services invocations of SSF are sequential and blocking, that is the basic call process is suspended. This is expressed in the specification by the synchronous invocation mode of the service `StartUp`. The OF-Class formalism enables the non-blocking and parallel invocations of services imported from different OF-Classes. So a scenario in which several points of control are involved can be described with the OF-Class formalism, i.e., an OF-Class `SSF` can invoke several OF-Classes `SCF` in a parallel and non-blocking way. In that case, several services can be invoked at the same time. If a precedence relationship must be respected between services, it is no longer expressed as an invocation priority, but as a priority of result collecting and processing. As for priority between invocations, the specification must state a priority policy for results collecting and processing. The proving tool is then able to validate that the policy does not exhibit interaction between services.

Code Generation

The OF-Class formalism is linked with another one called H-COSTAM which is mostly dedicated to code generation with optimization features. The translation from one to the other formalism is described in [Diagne96b]. The implementation of a tool for this translation is ongoing. So it will be possible to undertake code generation from an OF-Class specification. The couple of formalisms allows one to trace properties proved on the specification to the final implementation. The OF-Class formalism provides a very detailed specification level from which final code could be easily derived in an automated way. But we want to trace and verify properties via the H-COSTAM model.

4. Conclusion

This paper describes a proving toolset based on the coupling of the OO and PN paradigms and enabling a service designer to validate and verify a service specification. The proposed method can be incorporated in a standard software production environment without too much burden. The use of PN is implicit, that is the design engineer who models the new application does not need to know about PN theory and practise. It enables the formal verification of the applications in the earliest stage of their life-cycle. The shift from the OO initial model to the OF-Class model is easied by the design patterns. The transformation from OF-Class to OF-CPN is fully automated and supported by a tool integrated in an environment dedicated to formalization of software development along the life-cycle.

5. References

[Bouma94] L.G. Bouma & H. Velthuijsen, editors, «*Feature Interactions in Telecommunications Systems*», IOS Press, Amsterdam, 1994.

[Cameron94] E. J. Cameron et al, «*A Feature Interaction Benchmark in IN and Beyond*», In Feature Interactions in Telecommunications Systems, IOS Press, Amsterdam, 1994, PP1-23.

[Diagne96a] A. Diagne & P. Estraillier, «*Formal Specification and Design of Distributed Systems*», In Proc. 1st IFIP Int. Workshop FMOODS'96, Paris, France, March 1996.

[Diagne96b] A. Diagne & F. Kordon, «*A Multi-Formalisms Prototyping Approach from Conceptual Description to Implementation of Distributed Systems*», In Proc. 7th IEEE Int. Workshop on Rapid System Prototyping, Thessaloniki, Greece, June 1996.

[Valmari94] A. Valmari, «*Compositional Analysis of Place-bordered Subnets*», LNCS, vol. 815, PP531 -547.

[MARS94] MARS Team «*The CPN-AMI Environment version 1.3*», Laboratoire MASI, Institut Blaise Pascal, Université Pierre & Marie Curie