PILOTE: A Service Creation Environment in Next Generation Networks

Tatiana Aubonnet, Noëmie Simoni
ENST, 46 rue Barrault , 75634 Paris
taubonet, simoni@enst.fr

***Abstract*-We introduce a reference workbench architecture for UML (Unified Modeling Language)-based service creation founded on the decoupling of dynamic (Process) and static (Profile) aspects. This workbench has been developed in the PILOTE project (Processus d'Ingénierie du Logiciel des TElécommunications) of the French RNRT (Réseau National de Recherche en Télécommunications). A fairly high integration level among the tools of the workbench have been reached using metamodeling techniques (OMG standards MOF, XMI).**

This paper provides an example of Service Creation Environment in Next Generation Networks through a specification of the Pre-paid Calling Card service. These INCM-compliant service creation processes and profiles, being exported to a repository, are now easily re-usable. Considering these aspects and the obtained results, the PILOTE toolset seems to be a good environment for service creation in Next Generation Networks.

***Keywords*-UML, Service Creation Process, UML Profile, service specification, object-oriented modeling, INCM**

I. INTRODUCTION

Today, the creation of telecommunication services do thoroughly rely on off-the-shelf software engineering tool suites. These tools, to allow efficient and systematic service creation and evolution must therefore rely on well-defined standards. There is furthermore a need to automatically enact those processes. The PILOTE partners (Alcatel, Bouygues Telecom, ENST, France Telecom R&D, Softeam and Université de Nantes) are proposing a software development process which can be described as a set of workflows, each being executed within a so-called *working context* defined using UML Profiles [1].

The logical architecture of the PILOTE prototype is given in figure 1. Its fundamental goal is to clearly detach:

- the modeling of Process and Profiles,
- their storage in the Repository,
- and their use in the production.

This architecture also illustrates the separation between static (*Profile*) and dynamic (*Process*) aspects. A Profile is a work context that comprises a set of UML elements, a set of extensions that characterize the application field, and a set of rules of presentation, validation and transformation. A Process deals with the definition of tasks and activities (a sequence of tasks) and the declaration of actors engaged in those tasks.

This separation gave us a greater independence (static and dynamic) and consequently better opportunities to evolve as well as a more effective communication. Indeed, the static part (Profile) makes it possible to exchange specific field models, without modifying the choices of sequencing (*iterative*, *incremental* or *in cascade* process, for example).

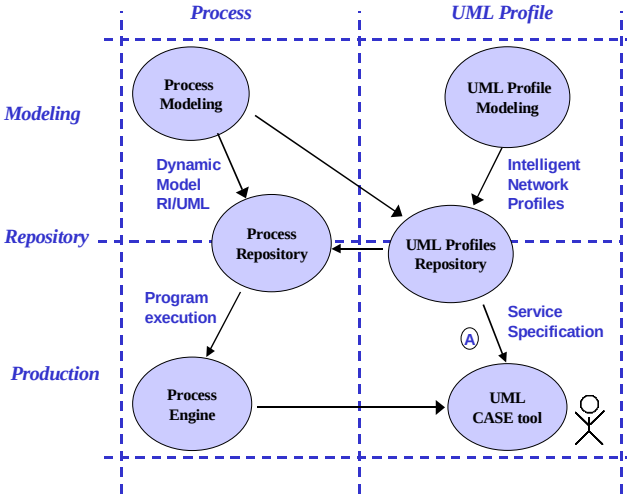

Fig. 1. PILOTE: Workbench Architecture

PILOTE eases re-utilisability and provides a guide for each stage of development. Process and profiles are exported to the repository. The process and profile sub-repositories promote knowledge sharing, exchange and re-use in accordance with the organization. The interface between the modeling, production and the repositories is done by an exchange of models based on the XMI/XML standards [6]. In the context of the PILOTE workbench, the role of repositories is to assure consistent communication between modeling and production tools during the service creation process.

We present in this paper the feedback of our experience on the PILOTE workbench, used as an IN (Intelligent Network) service creation environment (A, fig. 1). The specification of a Pre-paid Calling Card service is developed as an example.

The article will present in section 2 Workbench Architecture's modules of the PILOTE Service Creation Environment (Process, Profile and repositories) for IN. Before, paragraph A will outline the importance of the object-oriented approach applied in an IN context. Section 3 describes the assets of the PILOTE Service Creation Environment through a specification of the Pre-paid Calling Card service. Finally, we exhibit the advantages of the PILOTE Service Creation Environment in Next Generation Networks.

II. WORKBENCH ARCHITECTURE OF THE PILOTE SERVICE CREATION ENVIRONMENT

For the IN field we have an Intelligent Network Conceptual Model (INCM) (see fig. 2.). Each plane of the INCM corresponds to a level of abstraction of the capacities offered by the IN architecture. We will apply the key concepts of PILOTE : Process (paragraph B), Profile (paragraphs C, D) and repositories (paragraph E). Before, we describe the object-oriented approach for IN Services which differs from the procedural approach of IN standards [10].

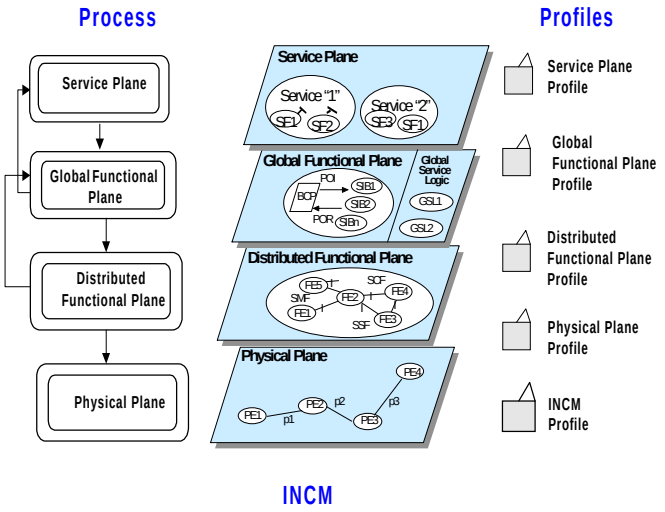

Fig 2. PILOTE: Application for Service Creation Environment

A. *IN evolution to Object Oriented Service definition*

A key objective of IN is to provide service-independent functions that can be used as service features to construct a variety of services. This allows easy specification and design of new services. A second objective is the provision of network-implementation-independent services. This approach isolates the services from the way the service-independent functions are actually implemented in various physical networks, thus providing services independent of underlying physical network infrastructures.

PILOTE adopted an *object-oriented approach*. Compared to the SIB approach, object-oriented methodology is considered a promising technique for service modeling. The main advantages of the object-oriented approach are:

- modeling the overall behaviour of the system,
- flexibility: object classes are independent, modular, portable, re-usable and easily extensible.

These objects form the basis for the definition of a Service Feature class library in the Service Plane [9]. The objects described in different INCM levels of abstraction can be generated by a transformation operation from the Service Plane.

B. *IN Process : the dynamic and specific part*

The process metamodel [15] specifies the basic concepts allowing us to describe the process workflows. An important requirement for the development of a support tool suite is to automatically enact the resulting workflows. This requirement implies restrictions on power of expression that will be offered to process modelers. A process consists of a set of tasks, activities, workers and workproducts, combined with a description of the temporal chaining between the tasks.

We propose the *IN Process* of service creation worked out in accordance with the one in force at Bouygues Telecom. Moreover, this example of IN process enabled us to define the elements of structuring (tasks and not activities) and to contribute to the process metamodel. We also modelled the process of service creation using a modeling tool of Alcatel (see fig. 3). This tool offers a graphical interface and makes it possible to define the elements of the process: tasks, products, roles; to specify the scheduling of tasks by means of decisions, synchronizations and transitions and to bind the tasks to the generated workproducts, the persons in charge and the work context (profile). We generate code (JMI, XMI) to export the process to the repository.

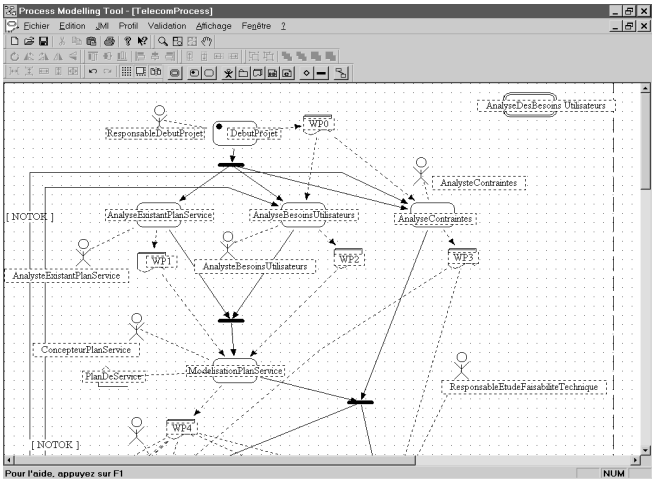

Fig. 3. An IN process modeling

C. *UML Profile Modeling*

Let us now look at the notion of the profile, as defined by the OMG (Object Management Group). Profiles already exist in UML 1.3, and are to be further developed in UML 1.4. But for now, we still have a “static” vision of profiles. We show that by associating behaviors to profiles, and by being able to link and combine profiles, we have a key tool in the handling of the development process using UML. UML 1.4 already plans to further integrate the UML profile notion with other UML elements, and to take into account improvements brought by the “action language”, that extends UML to describe dynamic processing [6].

UML Profiles are adapted to each domain. An environment is dedicated to the modeling of UML profiles: it contains UML extensions, validation rules, presentation rules and transformation rules. UML extensions can be used to adapt the notation to particular usage domains or development stages. Validation rules are used to check consistency criteria on a model for a given profile. In this way, it is possible to check at the end of analysis that all actors intervene in the Use Cases or that all Use Cases have some actor(s) associated to them. The quantity can by measured at each development stages. Presentation rules ensure that a good UML analysis diagram should present certain information and mask other information. Each stage has, therefore, its own UML presentation rules, which are used to filter diagrams or to create them automatically. Transformation rules are the development of workproducts or code generation rules which are used to automate development specifically for each field.

The notion of the UML Profile is already implemented in the PILOTE Case tool ("Objecteering") [5]. The modeling of the project's profile is a matter of software process engineering. The model of the profiles shows exactly which techniques are used, as the well as the way of working on a project.

D. IN Profiles : the static and exchange part

A *IN Profile* defines a work context by specifying, for a subset of the UML notation, a set of UML extensions that adapt this very notation to the individual needs of the Intelligent Network field. We re-use the three fundamental stages in the definition of IN profile which are:

- UML elements choice necessary for the modeling task.
- UML extensions definition in the field. The tool PILOTE makes it possible to associate the stereotype with an element by using the terms dictionary limited to the INCM field.
- Construction of presentation, validation and transformation rules.

UML elements are not all relevant to all service creation environments. UML elements used for Service Plane are: UseCase, Actor, Class, State, Transition and Activity.

- UseCase: represents a service using the collaboration of one or more IN Service Features.
- Actor: represents an external user of the system having a well-defined role in the Service Plane (customer, manager, provider of connectivity).
- Class: represents a Service Feature (ex: authentication, authorisation). Each Service Feature (SF) exists as an abstract class requiring a parameterization for each service.
- State: represents the states of the SF class.
- Transition: represents the transitions between states to a Class.
- Activity: represents an operation or an action executed out within a SF. An activity is interruptible.

The UML language can be customized using UML extensibility mechanisms: Stereotypes, Tagged Values and Constraints. UML defines stereotypes in order to extend the UML model elements semantics. The UML Profile Builder allows the new stereotypes creation under profiles, related to metaclasses, which have a name and an associated icon.

UML diagrams are the graphs that describe the contents in a view of the system. UML has nine different diagram types that are used in combination to provide all views of the service. UML diagrams proposed for Service Plane are: UseCase, Sequence, Activity, Class, State and Collaboration. Use Case diagram is used to represent the most important use cases involved with the current model. Use cases and actors are the main concepts in Use Case diagrams. Sequence diagrams are used to represent a co-operation among different objects. Instance and messages are the main concepts of this kind of diagram. Activity diagrams are special kinds of state machines that are used to model processes involving one or more classifiers. Its primary focus is to define sequences and conditions for the actions that are taken, rather than on which elements perform those actions. Class diagrams allow you to present the internal structure of an element and its relationships with other elements. State diagram represent the manner in which objects react to events. It is used to describe a state chart at a class level. Collaboration diagram represent the messages exchange between roles. Roles instances are the main concepts in this kind of diagram.

At each development stage, profiles are using rules for model validation, presentation and transformation. A set of rules establish the required models properties to be produced (validation rules), specify the contents of diagrams (presentation rules) and allow to automatically modify models or generate workproducts (transformation rules).

The PILOTE Service Creation tool uses the language J, the language for model manipulation of the Objecteering tool from Softeam. The profile modeler offers as well a graphical interface that helps to define model navigation, as filters and actions in order to create Intelligent Network Profile Rules.

We propose to identify a profile for each plane of the IN Conceptual Model (see fig. 2): Service Plane Profile, Global Function Plane Profile, Distributed Functional Plane Profile, Physical Plane Profile. A fifth profile, called INCM Profile, is used for global validation and tracability, it represents the inter-plane connections and the conformity in whole field. The five profiles of the Intelligent Network are available in the Repository under the XMI and JMI formats.

E. Repositories: hoarding of knowledge

The Process and UML Profile sub-repositories promote knowledge sharing and capitalize the experience of the definition of profiles and processes [3]. Components in a model are connected to the source code so that both model

and code can be re-used in different projects. In the PILOTE workbench context, the repositories assure consistent communication between modeling and production tools during the service creation process. The XMI exchange format is a very important feature for UML specifications. This standard [13] allows the effective exchange of process and profile models. The IN process (paragraph B) and five Profiles of the Intelligent Network are coded into XMI (JMI) and this code is available in a repository for rapid specification service.

III. PRODUCTION: SPECIFICATION PRE-PAID CALLING CARD SERVICE

This paper shows the assets of the PILOTE Service Creation Environment through *Pre-paid Calling Card* service specification. The Pre-paid Calling Card service enables a user to make a call from any telephone and for the call charge to be billed to the user's account, and which does not refer either to the calling line or to the called line. The Pre-paid Calling Card service specification using IN profiles has been very fast and efficient. In the next section, we will detail two practical points that prove our approach sound. We will indicate in this paragraph two practical points which have tested what we evoked in the preceding paragraphs: Use Case modeling and the transformations.

A. Use Case modeling

Use Case modeling is a technique used to decribe the functional requirements of a system. We propose to define Service Plane functions using Use Case diagrams. The Use Case diagram presented in Fig. 4 describes the essential use cases for the Pre-Paid Calling Card service. An actor symbolizes a external user and his relationships to an application model.

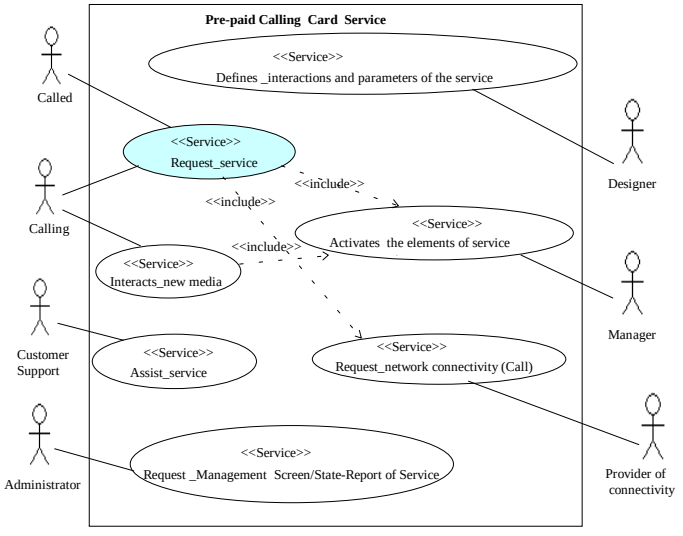

Fig. 4. Example of a use case diagram

The Use Case diagram represents the use case views of the system. These views are very important, as they affect all other views of the system. Both logical and physical architectures are influenced by the Use Case, because the specified functions (for example service interactions) in the Use Case model are implemented by those architectures.

A Transformations

The execution of the transformation rules makes possible the automatic production of parts of the model. It has enabled us to relate the various planes of the INCM, while preserving the coherence of the development. Let us take an example of transformation between the Service Plane (SP) and the Global Functional Plane (GFP).

The class diagram describes the static view of service in terms of classes (service features), modeling in object-oriented approach and relationships among the class. Identification of the service independent classes forms the basis for the definition of a service class library. Service independent classes are illustrated in figure 5 for the Pre-paid Calling Card service.

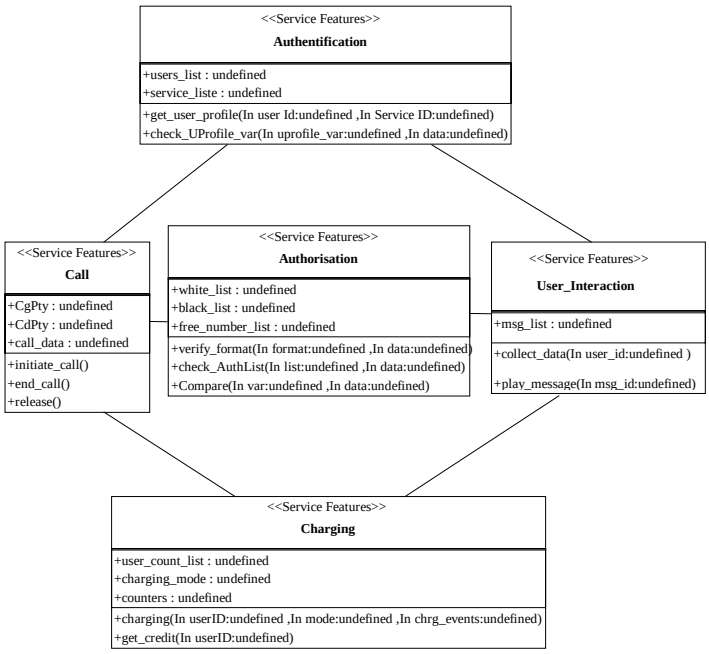

Fig.5. A Service Plane object diagram

The parameters (user ID and Service ID, see fig. 5) of an Service Plane operation (get_user_profile) become the attributes of the corresponding class (LSE, see fig. 6) on the Global Functional Plane. Our experiment in modeling Intelligent Network services using object-oriented methods allowed us to easily create re-usable software blocs called SIB macros-LSE (Logical Service Element). Our proposal for

INCM is that SIBs will no longer be used in future IN specifications to build the service creation environment.

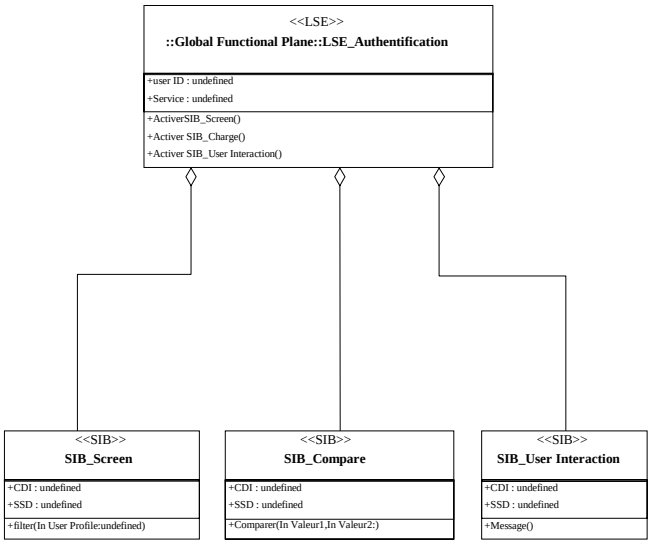

Fig. 6**.** A Global Functional Plane object diagram

The transformation model is one of most useful tools to automate application development. Indeed, it has enabled us to relate the various planes of the INCM, while preserving the coherence of the model.

CONCLUSION

We have presented an innovative approach to domain engineering based on the decomposition into workflow (Process) and work context (UML Profile). This approach has been assessed and refined based on experiences in service creation environment.

The PILOTE Service Creation Environment adopted an object oriented approach. The functional and operational requirements of Prepaid Calling Card service were identified. New object-oriented techniques such as UML provide an alternative way to disseminate IN services or applications into smaller parts or Object Classes and interfaces. Future IN specifications isolate the services from the way the service-independent functions are actually implemented in various physical networks, thus providing services independent of underlying physical network infrastructures.

The next wave of IN will permeate the information infrastructure and will involve the integration of flexibility and re-use. A flexible way to create and model services is essential if operators are to gain the competitive upper hand. PILOTE provide guides for each stage of development. The execution of the transformation rules make possible the automatic production of parts of the model; for example an object described in the Service Plane, can be generated on the others planes by a transformation operation.

We thus saw how it is possible to create UML profiles in Intelligent Network architecture by using UML case tool “Objecteering”. Besides, we managed to provide an interesting tool integration level using metamodeling techniques (OMG standards MOF, XMI, JMI).

Considering those aspects, the PILOTE toolset seems to be a good environment for service creation in Next Generation Networks.

ACKNOWLEDGMENTS

We would like to thank partners PILOTE: Yann Gaste and Gilles Désoblin (Alcatel), Luis Carroll and Monique Gibeaux (Bouygues Telecoms), Sylvie Vignes, Elie Najm, Khaled Sammoud, Alexandre Tauveron and Cyril Carrez (ENST), Martine Guerlus, Mariano Belaunde and Maria-Jose Presso (France Telecom R&D), Jerome Pequery, Gilbert Raymond and Philippe Desfray (Softeam) and Jean Bezivin (Universite de Nantes) for their participation, providing a reference model and support of the tool suite. We would also like to thank Arnaud Bailly (ENST) for his help and advice in finalizing this article.

REFERENCES

[1] M-J. Presso, G. Raymond, M. Belaunde, “PILOTE: A Tool Suite to Support UML-based Engineering Processes ”, *EDOC’2000* , 2000.
[2] RNRT (Réseau National de Recherche en Télécommunications). PILOTE Project, URL : http://www.telecom.gouv.fr/rnrt/
[3] PILOTE repository: http: // www.universalis.elibel.tm.fr
[4] G. Booch, J. Rumbaugh, I. Jacobson “The Unified Modeling Language User Guide”, Addison Wesley, 1999
[5] Softeam, Objecteering 4.3.1 User Manual / The J Language. 1999
[6] OMG UML Specification Version 1.3. “Have total control over your application development with UML”, Softeam 1999
[7] P809-GI “Mobility in the broadband environment based on IN evolution”, *EURESCOM*, 1999
[8] T. Aubonnet, N. Simoni “The management tools for an Architecture IN/TINA”, *GRESS*, Montreal 1999
[9] J. Zhang , N. Simoni, “Object-Oriented development for Intelligent Network with Distributed management”, *GLOBCOM’98*, 1998
[10] ITU-T Recommendation Q. 1222 "Service plane for Intelligent Network Capability Set 2", 1997
[11] ITU-T Recommendation Q. 1223 "Distributed functional plane for Intelligent Network Capability Set 2", 1997
[12] IN CS-4 Draft Baseline Document “Architecture Requirements for IN CS-4”, 1998
[13] OMG Specification, OMG “XML Metadata Interchange 1.0”, 1999
[14] OMG “Software Process Engineering Management Request for Proposals”, 1999
[15] OMG: Meta Object Facility. ad/97-08-14.
[16] OMG: Software Process Engineering Management Request for Proposals, 1999
[17] J-M. Cornily, M. Belaunde, “Specifying distributed object applications Using the Reference Model for Open Distributed Processing and the Unified Modeling Language”, *EDOC 99*, 1999.
[18] P. Desfray, “Hypergenericity: Automating Object Oriented development”, *Object Expert*, 1995.
[19] J. Rumbaugh, I. Jacobson, G. Booch, « The Unified Modeling Language Reference Manual », Addison Wesley, 1999
[20] PILOTE Project, URL : http://www.infres.enst.fr/pilote