Designing an Advanced Service Management Platform

Karim El-Khazen, Radu State
Motorola Labs
Centre de Recherche de Motorola (CRM)
91193 Gif-sur-Yvette, France
Email: {elkhazen, state}@crm.mot.com

ABSTRACT

The significant growth in the number of mobile users and their increasing demand for flexible access to various services has motivated significant research work in the area of future wireless access systems. It is assumed that heterogeneous radio access networks will be co-operating, as well as the IP-based backbone. Our assumptions on these different radio access technologies and the backbone result in a highly complex architecture. Consequently, management platform designers and developers need organizational guidance during the phase of creation of advanced service management platforms. Therefore, in this paper, a Unified Modeling Language process for designing such software platforms is presented.

KEYWORDS

Service Management, UML, Software Design

I. INTRODUCTION

As the wireless world evolves to become an ubiquitous communications infrastructure, there is a clear need for providing a detailed design methodology taking into account that future telecommunications platforms will be distributed over different technologies and operators. The systems are more and more real-time platforms, managing networks and services with high requirements. Moreover, the time granted for the service deployment is notably decreasing nowadays.

We present in this paper an approach towards designing an advanced service management platform based on the UML modeling. The Unified Modeling Language (UML) [2] is a standard notation for the modeling of real-world objects, providing a process for developing an object-oriented design methodology [5]. It gives a standard way to visualize, specify, construct and document the different phases of requirements, analysis, and design of complex software systems. Today's management platforms were developed using proprietary methodologies and solutions, and it is necessary to significantly adopt, as soon as possible, a well-defined and well-managed process for the development of future service and network management platforms.

This paper identifies the overall process that has been used in the development of an advanced service management platform. We modeled the system using Rational [6] software, and went through the different models such as the Use-Case, Analysis, Design, Deployment and Implementation models.

Our paper is structured as follows: Section II describes the context of our work. Section III introduces the multi-operator service management platform, whilst Section IV illustrates the software engineering approach for designing the above-mentioned platform. Finally, Section V concludes the paper and outlines future work.

II. PROBLEM STATEMENT

This design work has been done within the framework of the IST MONASIDRE project [4]. This approach considers the overall UMTS, DVB-T and Hiperlan-2 radio access technologies, while the fixed network is IP-based. This open management platform is capable of:

- Interworking with the Service Provider (SP) mechanisms,
- Monitoring and analyzing the performance of the managed infrastructure,
- And performing dynamic reconfigurations of the overall three types of networks.

Figure 1 identifies the different network domains and their interconnection with the distributed service and management platform.

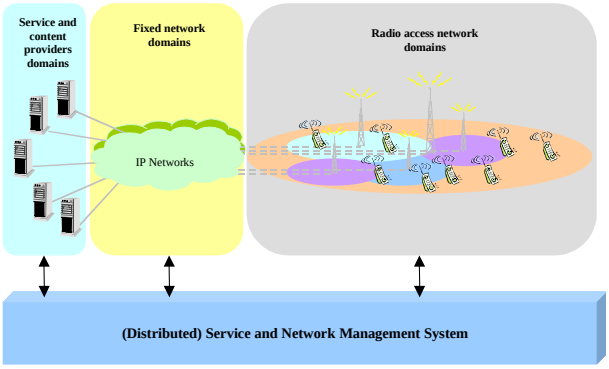

Figure 1

Before describing the design process, we will briefly present some of the requirements of the platform. First of all, the key characteristics of this integrated management platform are:

- Unified interface with the Service Providers,
- Development of a standardized interfaces for the resource reservation and the monitoring of the network elements of different radio access technologies,
- Incorporation of a set of sophisticated resource management strategies into an open software architecture,
- Integration of several management paradigms and protocols (SNMP [7], COPS [8], CMISE/CMIP [7]...)

Next, in the development of the MONASIDRE platform, a comprehensive process was needed for the following reasons:

- Distributed-oriented architecture: Common Object Request Broker Architecture (CORBA) is used as an underlying communication bus for services and application components,
- Multi-language implementation: C++, Java, HTML for web-based GUI, XML for flexible data representation…
- Multi-platform: MS Windows, Unix, Linux are used to support the global management platform,
- Partners with different characteristics cooperate in order to develop the software. Manufacturers, a network operator, a service provider, and universities co-work in this project with a common objective but different exploitations plans,
- Developments are performed in several physical locations.

III. THE MANAGEMENT ARCHITECTURE

The MONASIDRE system environment involves three independent network types (three different wireless access technologies), each with a MONASIDRE platform. Each wireless access system belongs to different operators. In this context, a pattern for distributing the software components of the management system is depicted in Figure 2.

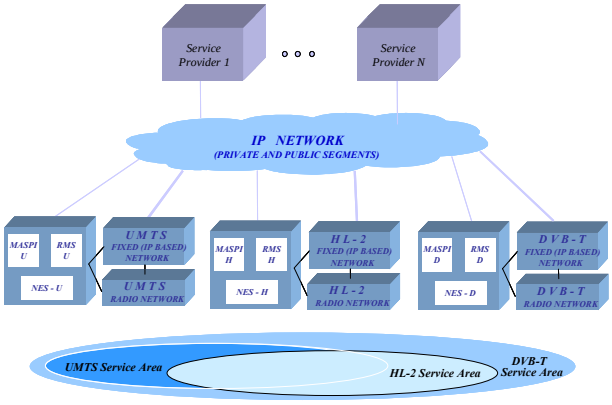

Figure 2

It is assumed that MONASIDRE components are distributed in each domain. The components in each domain are specialized for handling the specific (UMTS, HIPERLAN-2 and DVB-T) technology. However, these components can co-operate for handling service provider requests and/or new environment conditions. According to the above, Maspi(U), Rms(U), and Nes(U) are

components specialized for the UMTS network, similarly Maspi(H), Rms(H), and Nes(H) are components specialized for the HIPERLAN-2 network, and finally Maspi(D), Rms(D), and Nes(D) are components specialized for the DVB-T network.

The Network Environment Simulator (Nes) component of the MONASIDRE system is based on different engines, one for UMTS, one for HIPERLAN-2, one for DVB-T and one for the IP network.

All Maspi and Rms components have a common part, independent of the radio access technology "MaspiRati" and "RmsRati", as well as a part specific to the radio access technology they belong to, "MaspiRatd" and "RmsRatd". The following notations are used in this paper:

- MaspiRati: Maspi RAT Independent Part
- RmsRati: Rms RAT Independent Part
- MaspiRatd: Maspi RAT Dependent Part
- RmsRatd: Rms RAT Dependent Part

IV. PROPOSED APPROACH

We will describe the process adopted for the development of the MONASIDRE platform. This distributed architecture has been modeled taking into account its internal functionalities and its interactions with the surroundings.

Continuous development of quality software requires a predictable and repeatable process in order to be delivered on-time and on-budget. Moreover, it requires cohesive teamwork and a common understanding of the development tasks. All members of a software development team communicate through a common language. Coherent and sound processes enable us to develop individually, communicate collaboratively and deliver high-quality software.

A. Rational Unified Process

The Rational Unified Process (RUP) [3] provides a customizable framework, with all the templates, guidelines, and supporting scripts to do process customizations. We managed the tasks and responsibilities within the development organization. Its web-based interface makes it practical, especially in the case where the developers are spread around the world, like in MONASIDRE project.

Our project was based on several cycles. Each iteration cycle began with a plan for what should be accomplished and concluded with an evaluation of whether objectives have been met. A complete cycle comprises the following phases: Requirements, Analysis & Design, Implementation, Test, and Planning & Evaluation, as described in the Figure 3.

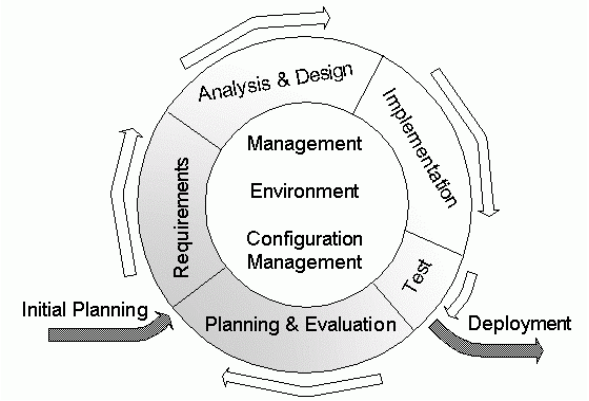

Figure 3

Moreover, RUP is tightly integrated with other modeling software, like Rational Rose. The later allowed us to gain the full benefits of the UML, by working, phase by phase, on the Use-Case, Analysis and Design, Deployment and Implementation models.

B. Use Case Model

A set of comprehensive use case diagrams were depicted during this phase of the project. For the sake of simplicity, we have not shown the whole set of diagrams. Use case diagrams added more power to the requirements gathering. They allowed us to validate scenarios and facilitated communication between analysts and users, and between analysts and clients.

C. Analysis and Design Model

As the system interacts with users and possibly with other systems (e.g. Service Provider), the objects that make up the MONASIDRE system go

through necessary changes to accommodate the interactions. The state diagram shows the states of an object and represents activities as arrows connecting the states. The activity diagram highlights the activities. State and activity diagrams describing particular scenarios/states of MONASIDRE are depicted in the Figure 4 and Figure 5.

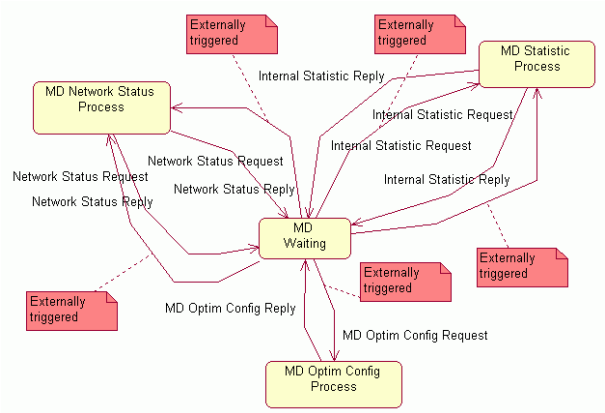

Figure 4

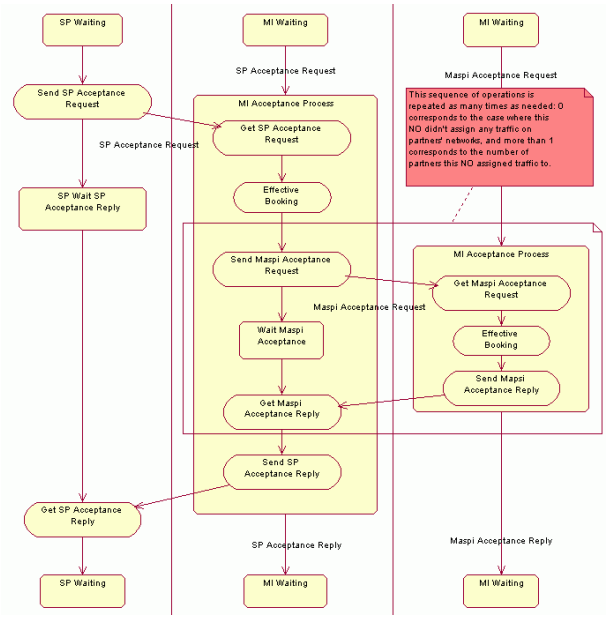

Figure 5

Once the use cases were identified and defined, we focused our attention on the class structure and object interactions that implement the system behavior. Class diagrams illustrate the static structure of objects and relationships in an object-oriented system. Out of 300 classes, forming the framework, an excerpt is shown in the Figure 6. This is the minimal sub-set of the set of classes that were defined within the project.

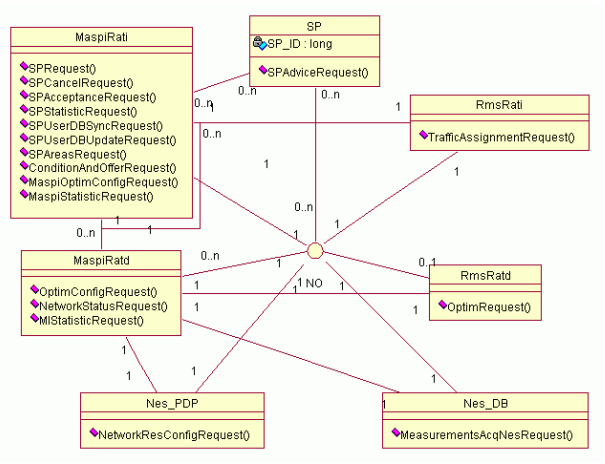

Figure 6

D. Deployment and Implementation Model

Component diagrams are one of the two kinds of diagrams found in modeling the physical aspects of object-oriented systems. They model the static implementation view of the system and are not only important for visualizing, specifying, and documenting it, but also for constructing executable systems through forward and reverse engineering. The definition of the components depends on the use of the components in the generated code in case of automatic generation of code. In the case of MONASIDRE, CORBA, C++, Java and XML code has been generated...

Deployment diagrams are the second type of diagrams used in modeling the physical aspects of systems. They involves the modeling of the topology of the hardware on which our system executes.

V. CONCLUSIONS

Developing with UML is fast and challenging; and the automatic code generation from UML decreases the development time and costs. Moreover, the high-level design and the UML implementation-independent characteristics allow software reusability. We were able to reuse initial building blocks for our platform during the integration phase.

The reverse engineering process which allowed us to convert existing source code into Rose elements, is essential to be able to model the complete system, including already existing packages (SNMP, COPS, XML…) written in C++ and/or Java.

We pointed out in this paper the key features that should characterize the design of an advanced management platform. The examined process is a step-by-step approach from problem definition to the realization of the platform. This is based on our experience in designing and implementing a "Beyond 3G" management platform. This UML modeling is aimed at ensuring consistent global delivery of software development services to reduce risk and guarantee quality.

Future work will consist in providing a set of design patterns needed for the design of advanced service management platform. These design patterns address several layers. At a software design layer, we should be able to identify patterns for a multi-developer interworking, thus providing reusable and sound approach for software design. At a framework architecture level, we need patterns particularly suited for object-oriented distributed and concurrent systems. Flexible and lightweight object composition within a larger system is of high interest in providing advanced service and network management platform.

ACKNOWLEDGEMENT

We thank Georges Martinez (Motorola Labs) for his comments and feedback. This work has been performed in the framework of the IST-2000-26144 MONASIDRE, which is partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues from National Technical University of Athens, Motorola Technology Centre Italy, Telefonica Investigacion y Desarrollo, Institut National des Télécommunications, University of Applied Sciences Valais, Shineline, Omnys Wireless Technology.

REFERENCES

[2] Object Management Group (OMG), http://www.omg.org

[3] Booch, Jacobson, Rumbaugh. The Unified Software Development Process. Addison Wesley. 1999.

|4] IST project MONASIDRE (Management Of Networks And Services In a Diversified Radio Environment), http://www.monasidre.com, Jan 2002

[5] Grady Booch. Object-Oriented Analysis and Design with Applications. Addison Wesley. 1994.

[6] Rational Software Corporation. http://www.rational.com

[7] Divakara K. Udupa. TMN Telecommunications Management Network. McGraw-Hill Telecommunications. 1999.

[8] RFC 2748. The COPS (Common Open Policy Service). IETF. 2000.