Wireless World Research Forum (WWRF)

Title of the Research Item

UML Process for Designing an Advanced Service Management Platform

Submitted to: WWRF WG3

Karim El-Khazen, Radu State

Motorola Labs
Centre de Recherche de Motorola (CRM)
91193 Gif-sur-Yvette
FRANCE

Subject Area: Network and Services Management

Objectives of the Required Research

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. As detailed in the "Network and Services Management" chapter of the Book of Visions [1], we can assume that UMTS, MBS and DBS will be three co-operating wireless access components, and that the core network will be IP-based. 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 Modelling Language (UML) [2] process for designing such software platforms is presented.

State of the Art in the Area

As the wireless world evolves to become an ubiquitous communications infrastructure, there is a clear need for providing a detailled 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.

UML is a standard notation for the modeling of real-world objects, providing a process for developing an object-oriented design methodology [5]. It gives us 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 which has been used in the development of a advanced service management platform. We modelled the system using Rational [6] software, and went throught the different models such as the Use-Case, Analysis, Design, Deployment and Implementation models. Different perspectives of our engineering model are now clearly described.

Possible Approach

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 with a common language. Rational softwares enable us to develop individually, communicate collaboratively and deliver better software.

- **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 manage the tasks and responsabilities within the development organization. Its web-based interface makes practical, especially in the case where the developers are spread around the world.

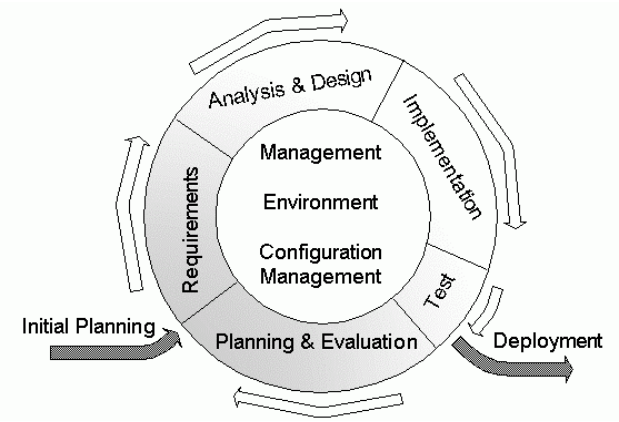

Figure 1

Each project iteration cycle begins with a plan for what will be accomplished and concludes with an evaluation of whether objectives have been met. A complete cycle comprises the following phases: Requirements, Analysis & Design, Implementation, Test, Planning & Evaluation, as described in the Figure 1.

- **MONASIDRE platform**

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 dynaminc reconfigurations of the overall three types of networks.

Figure 2 identifies the different network domains and their interconnection with the distributed service and management platform.

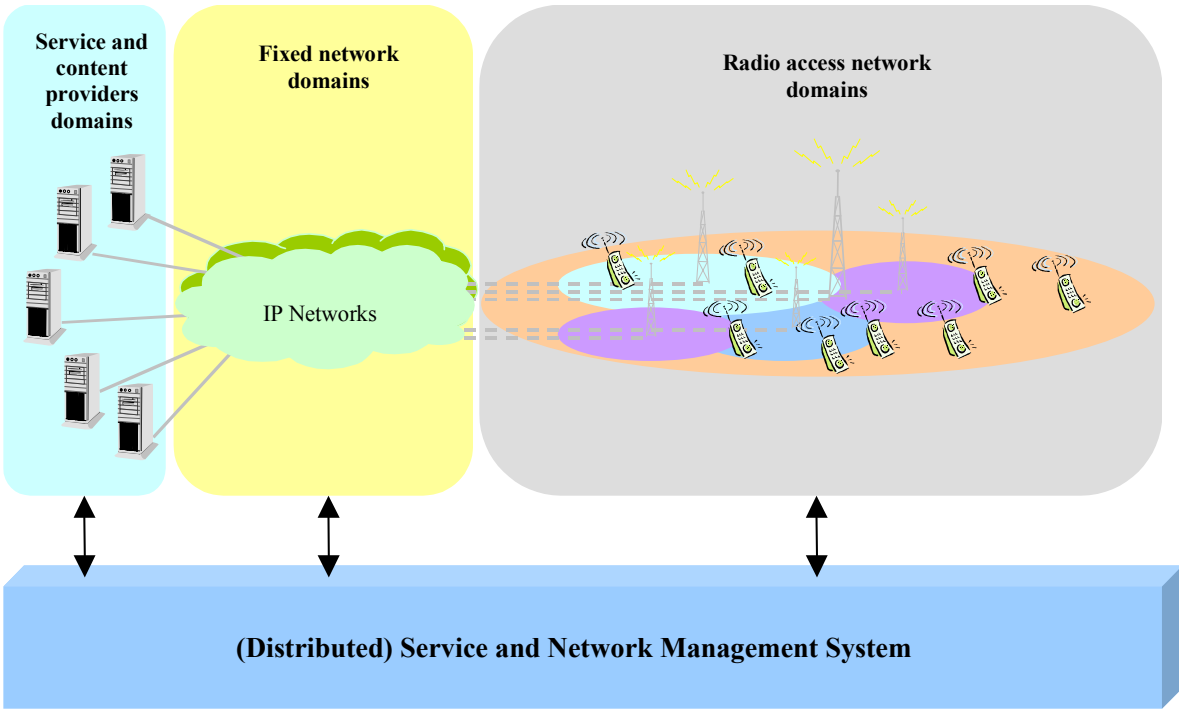

Figure 2

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 an 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 paradigm (SNMP, COPS, CMISE/CMIP...)

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

- Distributed architecture: Common Object Request Broker Architecture (CORBA),
- Multi-language implementation: C++, Java, HTML for web-based GUI, XML DTD…
- Multi-platform: MS Windows, Unix, Linux,
- Partners with different characteristics. Manufacturers, a network operator, a service provider, and universities co-operate in this project with a common objective but different exploitations plans,
- Developments performed in several physical locations.

This distributed architecture has been modelled taking into account its internal functionalities and its interactions with the surroundings.

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

- **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 add more power to the requirements gathering. They let us visualize use cases and facilitate communication between analysts and users, and between analysts and clients.

- **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 acommodate the interactions. The mechanism in UML which models changes is *State diagrams*. An *Activity diagram* is designed to be a simplified look at what happens during an operation or a process. It's an extension of the state diagram. The state diagram shows the states of an object and represents activities as arrows connecting the states. The activity diagram highlight the activities. State and Activity diagrams describing particular scenarios/states of MONASIDRE are depicted in the Figure 3.

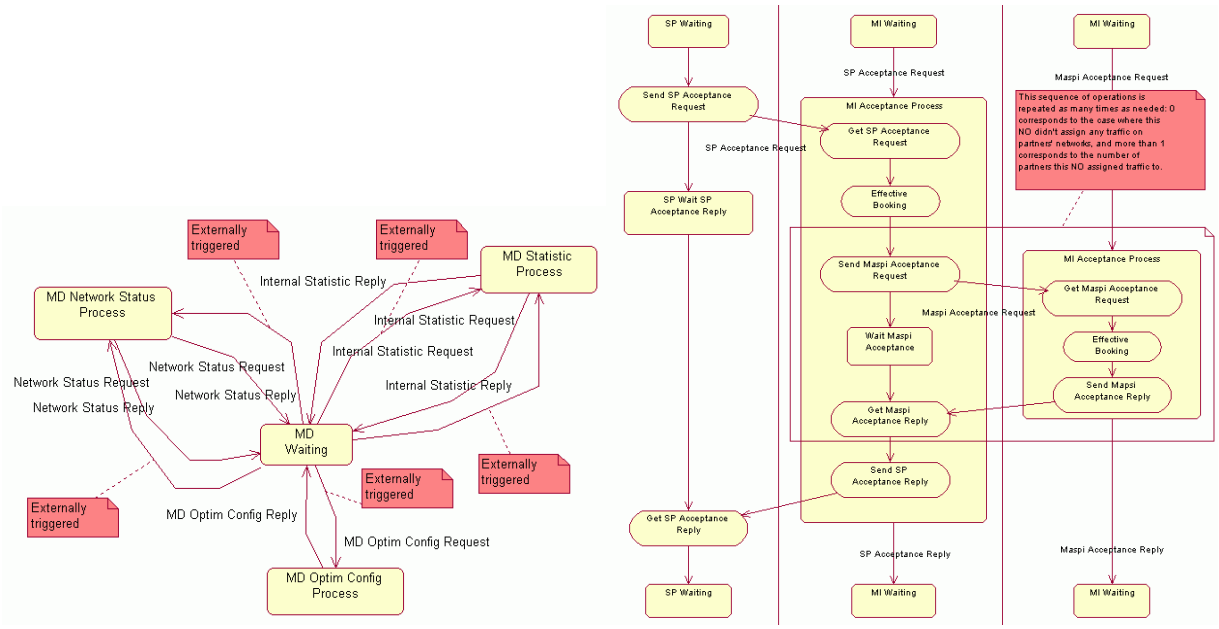

Figure 3

When the use cases of a project have been identified and defined, we had a good idea of what the MONASIDRE system does. Then 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. The main classes of the MONASIDRE platform are drawn in the Figure 4. This is the minimal sub-set of the set of classes that were defined within the project.

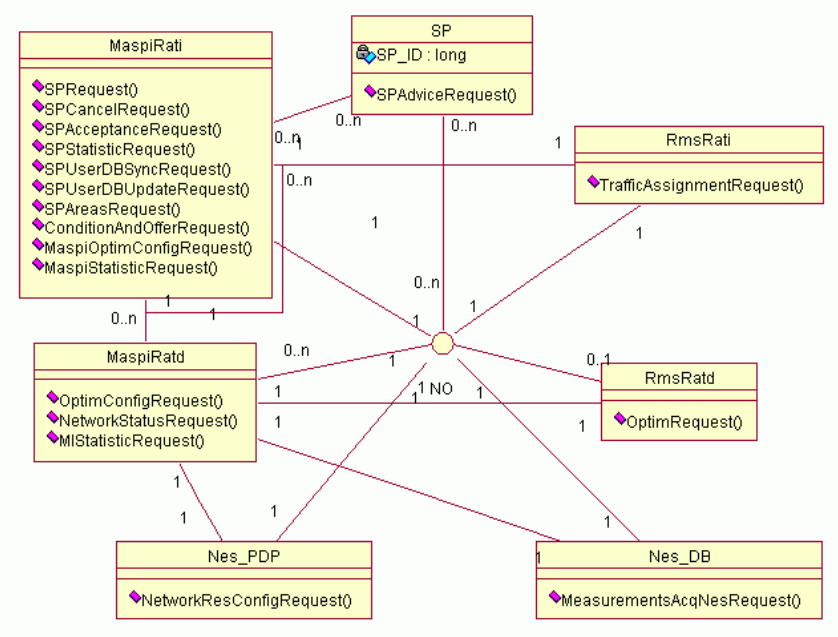

Figure 4

- **Deployment and Implementation Model**

Component diagrams are one of the two kinds of diagrams found in modelling 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, where CORBA code has been generated, only one component was defined in order to work one IDL file.

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

Expected results

Developing with UML is fast and simple; and the automatic code generation from UML description speeds up the coding process, and decrease the development time and costs. Moreover, the high-level design and the UML implementation-independent characteristics allow software reusability. The reverse engineering process which converts existing source code into Rose elements, is essential to be able to model the complete system, including already existing packages (SNMP, COPS…) written in C++ and/or Java.

We pointed out, in this paper, the key features that should characterise the design of an advanced management platform. The examined process is a step-by-step approach from problem definition to the realisation of the platform. This UML modelling is aimed at ensuring consistent global delivery of software development services to reduce risk and guarantee quality.

Acknowledgement

This work has been performed in the frame work 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 – Sequenza, Omnys Wireless Technology.

List of References

[1] WWRF Book of Vision
[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. 1994.
[6] Rational Software Corporation. http://www.rational.com