Chapter IX

Rendering Distributed Systems in UML

Patricia Lago
Politecnico di Torino, Italy

This chapter is about the use of UML in the design of distributed systems. We have been using UML in the context of both international and local projects, and we continuously face the need for additional constructs that are especially conceived for distributed software components in general, and specifically for software running on middleware such as CORBA.

In particular, it is our opinion that various UML diagrams would need some extensions (or customizations). Also we defined a development methodology supporting UML and a set of conventions that help us in overcoming such needs. Nevertheless, they contribute just to software documentation and communication, and do not provide any help in the low-level design phase as well as in the generation of code for the subsequent implementation phase. The chapter is intended to discuss our requirements and possible solutions.

INTRODUCTION

A software development methodology can be defined as “the overall approach to manage the development of software. It should incorporate an integrated set of processes, architecture rules, standards and practices, and should be supported by tools” (see the ESI dictionary of software). Processes should be the result of the know-how and expertise gained by a company with the development of various projects. Architecture rules should help in solving recurring architectural problems common to domain-specific applications, like standards provide stable patterns for different companies to integrate complementary (or similar) software products. At last, tools should guide the developer to carry out a production process, according to the supported methodology.

In other words, a software development methodology can be summarized as “a set of techniques supporting a defined development process, and a software product notation”: the process guides the developer in the various phases of software engineering, whereas the notation supports the visual description of the resulting software product.
While the development process varies according to at least the type and dimension of the system to be developed, and the dimension and organization of the developing team, UML represents a *de facto* standard notation for object-oriented software modeling (OMG, 1997). Nonetheless, for the development of domain-specific software systems, like distributed systems, it still needs further work. In particular, in this chapter we focus on the specification of those aspects that are critical to distributed software architectures. Our aim is to describe both the experience gained in modeling such systems, and the requirements on a notation suitable for their complete representation.

In particular we based this work on developments carried out in the field of advanced telecommunication services (or service architectures), for which aspects like inter-operability, portability, software distribution and deployment need specific attention. Hence, to describe a service architecture thoroughly, there is the need for a modeling language with constructs describing the various aspects mentioned above, in an integrated and possibly intuitive way.

The work is structured as follows: “Background” discusses related work. “Impact of Distribution on UML Diagrams” describes those aspects that are critical in representing distributed systems: for each aspect, the associated UML diagram is discussed to draw the lacks and possible extensions. “A Development Methodology for Distributed Software” draws a methodology for software development, in which discussed diagrams find context. The chapter ends with some Conclusions and future trends.

**BACKGROUND**

This work addresses two directions in the development of distributed object-oriented systems: the notation used for modeling and the process followed by software engineers. This Section discusses some related work and makes a comparison with the work presented here.

**Notations**

UML is the successor of the object-oriented analysis and design notations of the late '80s and the early '90s. In particular it is an evolution of Booch & Rumbaugh’s OMT, Jacobson’s OOSE, and other object-oriented methods like Harel’s statecharts.

In this context we do not want to discuss the benefits and drawbacks of the different modelling notations for which comparisons and surveys already exist in the literature (see for instance Berard, 1995; Wieringa, 1998). We base this on the fact that UML was a standard *de facto* since some years earlier, and since 1997 it is an OMG standard. Rather, we would like to discuss some examples of distributed systems modelling, which face problems in using UML.

Emmerich (2000, pp. 50-51) discusses some differences in the development of non-distributed applications and distributed objects. For instance, in the first case object creation is considered an implementation problem, whereas in distributed systems it is a design problem that needs dedicated modelling. Also, object composition, which represents a critical aspect in distributed systems, can be represented in UML, e.g., by using packages in Class diagrams, even if the notation syntax lacks in expressiveness and communication. Similar conclusions are discussed in our work.

To model software architectures for telecommunication services, both ISO (ISO/IEC,
Related Content

Using the Semantic Web Rule Language in the Development of Ontology-Driven Applications
Martin O’Connor, Mark Musen and Amar Das (2009). Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches (pp. 525-539).
www.irma-international.org/chapter/using-semantic-web-rule-language/35873/

A Generic Framework for Defining Domain-Specific Models
www.irma-international.org/chapter/generic-framework-defining-domain-specific/30535/

Rules Verification and Validation
www.irma-international.org/chapter/rules-verification-validation/35863/

Sharing Ontologies and Rules Using Model Transformations
www.irma-international.org/chapter/sharing-ontologies-rules-using-model/35871/

Modular Rule-Based Programming in 2APL
www.irma-international.org/chapter/modular-rule-based-programming-2apl/35853/