Distributed Systems: From Models to Components

Fabrice Dubois[1], Marc Born[2], Harald Böhme[3], Joachim Fischer[3], Eckardt Holz[3], Olaf Kath[3], Bertram Neubauer[3], and Frank Stoinski[3]

[1] France Telecom R&D, 2, Avenue Pierre Marzin
22300 Lannion, France
fabrice.dubois@rd.francetelecom.fr
[2] GMD FOKUS, Kaiserin-Augusta-Allee 31,
D-10589 Berlin, Germany
born@fokus.gmd.de
[3] Humboldt-Universität zu Berlin, Dept. of Computer Science,
Rudower Chaussee 25, D-12489 Berlin, Germany
{boehme|fischer|holz|kath|neubauer|stoinski}@informatik.hu-berlin.de

Abstract. Advanced design methods are needed to fulfill the increasing requirements of telecommunication service development. For a design method the relevant concepts for the application domain have to be defined, a supporting notation has to be declared and finally rules have to be developed to map design models to supporting runtime environments. The ITU-T has followed this route by defining concepts for the design of distributed telecommunication applications and supporting notations for these concepts. In the past, the ITU-T has defined several languages and notations to support structural and behavioral descriptions of distributed telecommunication systems, namely ODL, SDL-2000 and MSC-2000. With the rise of the component age, an additional technique (DCL) is under development that enables component based manufacturing of distributed systems. Beside these languages, the ITU-T recognized the common need for open, component aware object middleware platform standards as the runtime environment for these systems. This contribution is about integration.

1 Motivation and Introduction

A dedicated and efficient design methodology contributes significantly to a reduction of the time to market distributed applications and telecommunication services. An appropriate treatment of all kinds of communication aspects lies in the very nature of the targeted application domain. These aspects span from functional requirements on object interactions over quality-of-service issues to security properties. Taking into account the broad acceptance of object middleware technology, middleware platforms provide an ideal implementation environment for such designs.

Conceptually, an appropriate design method may be split into three separate parts:

R. Reed and J. Reed (Eds.): SDL 2001, LNCS 2078, pp. 250–267, 2001.

- A concept space, that contains all relevant entities that conceptually reflect the elements of the problem domain and the information for the description of these elements and their relations;
- a concrete notation to visually specify models using the elements of the concept space;
- and a set of rules to specify the mapping of models onto middleware technologies.

Following this approach, a concept space is independent of a specific design notation. Design models can be developed in different notations but are based on the same concepts. Design information can then be exchanged on the basis of the common concept space. Second, both the notation and the concept space are independent of a specific runtime-environment. The same design can be mapped onto different environments. This enables a high flexibility and is also important for the aspect of re-use of component design models.

In the past, the ITU-languages ODL [1] and SDL [3] as well as the proposed language DCL [19] represented different aspects within the design of distributed systems. No common concept space for them has been defined, an integration is only possible by pair-wise direct mappings based on notational concepts ([12,13]). This is also valid for the relation of ITU languages to languages of other communities, for instance between ODL and OMG-Component IDL or SDL/MSC and UML [5].

This paper discusses the concepts of and the relations between selected ITU-languages and the current trends for their future development and refinement taking into account corresponding OMG activities. In order to apply different techniques with their specific advantages in a combined approach for the design of distributed systems, the underlying concepts have to be integrated within a common concept space. This can be achieved in 3 main steps:

- the identification of those concepts of each technique, that are relevant to a combined approach;
- the definition of the semantics for a combined concept space;
- the realization of the integration within a common meta-model framework by applying a suitable technology.

Consequently, major parts of the components that form a distributed system will be generated from design models that are based on the common meta-model and target specific object middleware platforms. Therefore, an example of such a platform will be presented here as well.

2 Modelling Distributed Systems

2.1 ODL for Modelling of Software Components

Starting from the basic reference model of Open Distributed Processing (ODP), the TINA community has taken up this general approach to define dedicated computational modelling concepts and a supporting notation Object Definition