Chapter 16

Service–Oriented Development of Fault Tolerant Communicating Systems: Refinement Approach

Linas Laibinis
Åbo Akademi University, Finland

Elena Troubitsyna
Åbo Akademi University, Finland

Sari Leppänen
Nokia Research Center, Finland

ABSTRACT

Telecommunication systems must have a high degree of availability, that is, a high probability of correct and timely provision of requested services. To achieve this, correctness of software for such systems should be ensured. Application of formal methods helps increase confidence in building correct software. However, to be used in practice, formal methods should be well integrated into existing development process. In this paper, the authors propose a formal model-driven approach to development of communicating systems. The authors formalize and extend the Lyra approach—a top-down service-oriented method for development of communicating systems. Lyra is based on transformation and decomposition of models expressed in UML2. The authors formalize Lyra in the B Method by proposing a set of formal specification and refinement patterns reflecting the essential models and transformations of the Lyra phases. Moreover, this paper extends Lyra to integrate reasoning about fault tolerance in the entire development flow.

DOI: 10.4018/978-1-4666-0912-9.ch016
1 INTRODUCTION

Modern telecommunication systems are usually distributed software-intensive systems providing a large variety of services to their users. Development of software for such systems is inherently complex and error prone. However, software failures might lead to unavailability or incorrect provision of system services, which in turn could incur significant financial losses. Hence it is important to guarantee correctness of software for telecommunication systems.

Formal methods have been traditionally used for reasoning about software correctness. Nevertheless, they are yet insufficiently well integrated into current development practice. Unlike formal methods, Unified Modeling Language (UML) developed by Rumbaugh, Jakobson, and Booch (1998) has a lower degree of rigor for reasoning about software correctness but is widely accepted in industry. UML is a general purpose modelling language and, to be used effectively, should be tailored to a specific application domain.

Nokia Research Center has developed the design method Lyra described by Leppänen, Turunen, and Oliver (2004) – a UML2-based service-oriented method specific to the domain of communicating systems and communication protocols. The design flow of Lyra is based on the concepts of decomposition and preservation of the externally observable behaviour. The system behaviour is modularised and organized into hierarchical layers according to the external communication and related interfaces. It allows the designers to derive the distributed network architecture from the functional system requirements via a number of model transformations.

From the beginning Lyra has been developed in such a way that it would be possible to bring formal methods (such as program refinement, model checking, model-based testing etc.) into more extensive industrial use. A formalization of the Lyra development would allow us to ensure correctness of system design via automatic and formally verified construction. The achievement of such a formalization would be considered as significant added value for industry.

In this paper we propose a set of formal specification and refinement patterns reflecting the essential models and transformations of Lyra. Our approach is based on stepwise refinement of a formal system model in the B Method proposed by Abrial (1996) – a formal framework with automatic tool support. While developing a system by refinement, we start from an abstract specification and gradually incorporate implementation details into it until executable code is obtained. While formalizing Lyra, we single out a generic concept of a communicating service component and propose patterns for specifying and refining it. In the refinement process the service component is decomposed into a set of service components of smaller granularity specified according to the proposed pattern. Moreover, we demonstrate that the process of distributing service components between different network elements can also be captured by the notion of refinement.

To achieve system fault tolerance, we extend Lyra to integrate modelling of fault tolerance mechanisms into the entire development flow. We demonstrate how to formally specify error recovery by rollbacks as well as reason about error recovery termination.

The proposed formal specification and development patterns establish a background for automatic generation of formal specifications from UML2 models and expressing model transformations as refinement steps. Via automation of the UML2-based Lyra design flow we aim at smooth incorporation of formal methods into existing development practice.