Chapter XI

XML-Based Analysis of UML Models for Critical Systems Development

Jan Jürjens, TU München, Germany
Pasha Shabalin, TU München, Germany

Abstract

High-quality development of critical systems poses serious challenges. Formal methods have been proposed to address them, but their use in industry is not as widespread as originally hoped. This chapter proposes to use the Unified Modeling Language (UML), the de-facto industry standard specification language, as a notation together with a formally based tool-support for critical systems development. The authors extend the UML notation with new constructs for describing criticality requirements and relevant system properties, and introduce their formalization in the context of the UML executable semantics. Furthermore tool-support concepts for this approach are presented, which facilitate transfer of the methodology to industrial applications.

Introduction

Modern society relies on distributed IT-based infrastructures in many aspects including communication, finance, energy mining and distribution, and transportation. The disrup-
tion or incorrect functioning of these systems may threaten the economical or even physical well-being of people and organizations.

Complex distributed systems can have subtle flaws, which are not obvious and often cannot be detected by common testing procedures. Additionally, the distributed character of such infrastructures and the interconnection of modern information systems make remote and anonymous attacks on them possible. Examples indicating the potential scale of the problem include the following:

- The survey published in 2002 by Computer Security Institute in cooperation with FBI indicated that 90% of the interviewed organizations detected intrusion into their IT infrastructures within the last year. Only 44% of them were willing and able to quantify their losses. These 223 companies reported $455,848,000 in financial losses (Richardson, 2003).
- In 1997, a NASA hacker team broke into the U.S. Department of Defense and U.S. electric power grid system networks. They were able to provoke power outages and 911 emergency phone overloads in Washington, D.C. (Schneider, 1999).
- Spectacular examples for software failures in complex systems include problems with the Ariane 5 rockets: an independent inquiry board set up to investigate the explosive failure in 1997 reported that the flight control system failed because of errors in computer software design.

Obviously, the problem with critical systems was not left unnoticed, and many methodologies exist to improve their reliability. However, as we argue in the following section, we are still far from a satisfying solution. In this chapter, we would like to demonstrate how the Unified Modeling Language (UML) together with XML-based processing of UML models, offers a significant step toward a solution to the problem, in the context of model-based development of critical systems using UML.

**Overview and Background**

Traditionally, different methods exist for ensuring reliability of critical systems:

**Break-And-Fix.** This approach accepts that deployed systems may fail; whenever a problem is noticed and identified, the error is fixed. The Break-And-Fix approach is probably the most obvious one, however, it has many drawbacks. It is inherently disruptive — fixing the system often implies distributing patches, which disturbs users, annoys customers, and destroys their confidence. What is worse, the method is unsafe and insecure — we can never be sure that the new problem will not disturb critical functionality, or that it will not be spotted at first by a malicious person, who will try to compromise the system further.

**Traditional formal methods**, on the other hand, offer very good quality of the developed critical systems. There is much successful research in this direction. For security-