Evaluating Dependability Attributes of Component-Based Specifications

Ivica Crnkovic\textsuperscript{1}, Lars Grunske\textsuperscript{2}

\textsuperscript{1}Mälardalen University, Department of Computer Science and Electronics
Box 883, 721 23 Västerås, Sweden

\textsuperscript{2}School of ITEE, ARC Centre for Complex Systems, University of Queensland,
4072 Brisbane (St.Lucia), Australia

\textsuperscript{1}ivica.crnkovic@mdh.se, \textsuperscript{2}grunske@itee.uq.edu.au

Abstract

Component-Based Development (CBD) is established in many application domains. There is a strong trend in applying the same approach in different domains of dependable systems. However, a precondition of a successful use of CBD in these domains is the utilization of theories, methods and technologies to predict and evaluate dependability attributes. This tutorial gives an analysis of current methodologies of attribute-specific evaluation methods for dependable component-based systems. We identify limitations of the current technologies, discusses existing and possible new solutions to overcome these limitations both from a research-oriented and practical perspective.

1. Introduction

Inspired by other engineering disciplines, Component-Based Software Engineering (CBSE) \cite{4,8} aims at building systems from pre-existing components and building components that can be reused in different systems and in this way achieve development that is more efficient.

CBD has been very successful in building software systems in many domains, but has been less utilized in development of dependable systems \cite{1}, in particular software-intensive embedded systems. There are several reasons for that: (i) different constraints (such as resource consumptions) and dependability requirements have not been concerns of component-based technologies; (ii) some basic principles of CBD (encapsulation and information hiding, extensibility, bottom-up approach) make it more difficult to achieve dependability properties required for such systems.

Today we witness an increasing effort in research and in the industry to combine CBD with modeling and analysis methods and tools for dependable systems. Actually, many modeling and analysis theories and tools exist, but are not integrated in CBD. For this reason, the most important initiative is to bring these worlds together and make researchers and practitioners aware of both sides.

2. Basic principles

The interest in CBSE is strongly reflected in new and innovative software technologies, but it is also based on a long history of work in modular systems, structured design, and most recently in object-oriented systems. CBSE has further developed the concept of interface as a specification of components extending it from a pure syntax definition to separation of provided and required, contractually-based and semantically specified interface. Further, CBSE has established certain principles of development: reusability, substitutability, expandability and composition. Different component technologies have implemented these principles to different extents. These principles will be discussed in the tutorial.

In classical engineering disciplines such as civil engineering, the component-based approach is the fundamental design principle: systems are constructed from (existing) components. This is also the case with embedded software; in complex embedded systems, where components are usually nodes consisting of sensors, actuators and control computing systems. Software is embedded in these nodes and is not treated separately. This approach works fine as long as the nodes are relatively independent of each other, and as long as the communication between them is precisely defined. By increasing functionality, this communication significantly increases, and the functional implementations are not localized on nodes. Instead, services become identified as independent units (i.e. components) of the systems. This wipes out the clear border of hardware components and differentiates between hardware and software parts. In addition, due to flexibility and low cost requirements, there are demands for separation of software development from hardware development. Services become implemented as software components that are performing on an underlying hardware. In this tutorial, we illustrate these differences through some industrial examples (from the automotive and telecommunication industry).
Dependability is related to trustworthiness (i.e. assurance that a system will perform as expected) and to some extent to survivability (i.e. capability to fulfill its mission in a timely manner). Dependability is a crucial attribute for safety-critical systems (i.e. systems for which failure may have catastrophic consequences), but also for the mission-critical, and business-critical systems. The requirements for dependability are increasingly important for general-purpose everyday-used systems. Dependability is an increased concern in software development in general, as well as in CBD. Dependability is characterized by a set of attributes: availability, reliability, integrity, confidentiality, safety and maintainability. We should add here security (that is related to confidentiality). Most of these properties are of interest for software-intensive systems but also for a wide range of software systems. For embedded systems, we also have demands on resource utilization and real-time properties. Since CBD systems are built from components, their specification and their compositions must be related to evaluation and composition models of the properties of interest.

3. Composability and Analysis

In a bottom-up approach, which is typical for building systems from existing components, composition of components and composition of their properties are important concerns in predicting properties of the systems. While compositions of functional properties, although difficult, are computable, and consequently predictable, compositions of non-functional properties are significantly more difficult to model. In many cases properties are not even composable. In this tutorial, we shall present a classification of properties that distinguishes properties according to their composability [4].

In the cases in which it is not possible to reason about a composition, different evaluation and analysis methods on the system level must be applied. These methods are based on different evaluation and analysis theories, component specifications and system architecture. Dependability evaluation methods for component-based systems are currently focused towards specific dependability attributes.

Reliability Evaluation. Reliability is defined as the ability of a system to operate correctly according to its specification for a given time interval. To be able to analyze the system’s reliability, the reliability of component services must be defined in relation to the deployment context and the reliability of required services [5,7].

Safety Evaluation. The aim of safety evaluation techniques is to determine whether a specification and the resulting system can meet its safety requirements, where a safety requirement is a description of a hazard combined with the tolerable probability of this hazard. Common safety evaluation methods are based on different fault propagation notations that can be applied to estimate hazard probabilities of component-based architectures [6].

Real-time Evaluation. Real-time properties are related to time-related system requirements such as system response, guarantee of a successful completion of an operation and similar. When applying CBD, questions such as transformation of components to execution entities (tasks or threads), component specification of real-time properties, and composition of real-time properties become important issues to achieve system temporal correctness.

Performance Evaluation. Performance evaluation techniques are used to predict if software systems can satisfy user performance goals such as mean reaction times on external stimuli or number of possible simultaneous requests [2]. Common performance evaluation model to perform these predictions are Queuing Networks, Petri Nets and Markov Models.