An Automated Dependability Analysis Method for COTS-Based Systems

Lars Grunske[1] and Bernhard Kaiser[2]

[1] School of Information Technology and Electrical Engineering ITEE,
University of Queensland, Brisbane, QLD 4072
grunske@itee.uq.edu.au
[2] Department of Software Engineering and Quality Management,
Hasso-Plattner-Institute for Software Systems Engineering,
University of Potsdam, Prof.-Dr.-Helmert-Straße 2-3,
D-14482 Potsdam, Germany
bernhard.kaiser@hpi.uni-potsdam.de

Abstract. The increasing application of COTS-components and component-based software engineering has entailed the development of appropriate component specifications. In the embedded systems domain it would be desirable to benefit from these component specifications to integrate and automate safety and reliability analysis. For this reason, we propose in this paper a component-based dependability analysis technique that annotates components with failure mode assumptions. The probabilities and dependencies of these failure modes are specified by Component Fault Trees (CFT's). Based on these CFT's and the architectural model the propagation of failures throughout the system can be automatically determined and a quantitative analysis is possible.

1 Introduction

Constructing software systems with reusable or COTS-components has become a popular approach for several reasons, including cost reduction, quality improvement and shorter time to market. Moreover, humans are incapable of handling highly complex systems without decomposing them.

Another predominant paradigm in modern software development is model-based development, because it facilitates the development of complex systems and supports their decomposition: modeling techniques such as ROOM [16] or some UML 2.0 models reflect the component structure and depict communication mechanisms between the components. Models have been unified and integrated to cover all development phases from requirements analysis to code generation and testing. Safety and reliability analysis however have not yet been integrated with the other phases, mainly due to their different modeling approaches. Safety or reliability cases must be built from scratch, causing additional workload and compromises the consistency of the analyses with the actual system. Reusable component dependability models and their automatic integration according to the system structure would facilitate an integrated development process.

X. Franch and D. Port (Eds.): ICCBSS 2005, LNCS 3412, pp. 178 – 190, 2005.

As a solution to this issue, we present a method that annotates component models with Component Fault Trees (CFTs). These CFTs describe how failure modes of the incoming messages together with internal faults of the components propagate to failure modes of the outgoing messages. Since the interconnection of the components of a system by their ports is described in the structure diagram, we can compose these CFTs automatically and perform the quantitative analysis on the system-level Fault Tree.

The rest of the paper is organized as follows: In Section 2, the basics of the component-based models and of CFTs are introduced. In Section 3, we explain the proceeding of a component-based dependability analysis in detail. The case study of a protection system in Section 4 demonstrates the application in practice and in Section 5 we present the safety analysis tools BALANCE and UWG3 that support the method. We conclude with a survey of related work in Section 6 and a summary in Section 7.

2 Preliminaries

2.1 Component Based Software Engineering

Building a software system with self-contained and exchangeable components is a precondition for efficient modeling and reuse, which are both key elements of mature engineering disciplines. Thus, many current design approaches divide systems recursively into components (sometimes called capsules), which are instances of component-classes. Each component-class is described by an appropriate set of models, i.e. structure and behavioral models. A component-class can either be flat, i.e. not supposed to be refined any further, or contain subcomponents. In the latter case, the component is called a hierarchical component of which the entire system is a special case.

Components are encapsulated entities that hide their internal details and communication with their environment is only possible via ports. The selected model implicitly determines the kind of information that is transferred via ports; examples are discrete event signals, continuous data streams, or any kind of service requests and the corresponding responses. In many models it is allowed that different messages or services are transmitted across the same port. Ports or their associated services can have a direction (in or out). In this case, they must be connected as complementary pairs (input to output, service provider to service consumer etc.). The associated semantics is that information flows from a source component to a target component or that a service is required by a client component and provided by a server component.

The architecture of the system, graphically depicted by the structure diagram, specifies how a higher-level component is built of lower-level components and how these can interact during the runtime of the system. Therefore it must be described which ports of the components must be connected. Two basic connection mechanisms can be distinguished, connection and binding [6]. The difference is that a connection interconnects two ports on the same hierarchical level, whereas a binding interconnects two ports in different hierarchical levels, i.e. a port of a subcomponent with a port of its enclosing component [6]. As an example Figure 1 depicts the structure