DISSERTATION DIRK NIEBUHR

DEPENDABLE DYNAMIC ADAPTIVE SYSTEMS

– APPROACH, MODEL, AND INFRASTRUCTURE

ABSTRACT

Today's Component-based systems tend to be more and more dynamic. Due to the increased mobility of devices hosting Components, Components are entering respectively leaving a system at runtime. Therefore, a system's Component Binding, which is part of the System Configuration, needs to be changeable at runtime.

This dynamic adaptation of the System Configuration imposes Dependability issues, which are hard to address, since Components which need to interact in a system are not necessarily known to the system up-front. Therefore, it is not possible to determine a semantically compatible System Configuration in general.

In this thesis we will discuss approaches how to achieve Dependable Dynamic Adaptive Systems which are bound from unknown Dependable Dynamic Adaptive Components at runtime. These approaches are based on a formal model for Dependable Dynamic Adaptive Systems. We come up with a feasible solution to establish Dependable Dynamic Adaptive Systems, which is based on runtime testing.

Although this approach does not provide statements about the system, that are comparable with statements derived from formal verification, it is applicable in Dependable Dynamic Adaptive Systems at runtime of a system. It provides runtime-statements, whether a Component Binding is incompatible, *before* such an incompatible binding can lead to problems during system execution.

We will motivate and describe our approach at an application example from the emergency management domain. The infrastructure DAiSI – **D**ependable Dynamic **A**dapt**i**ve **S**ystem **I**nfrastructure – has been realized within this thesis. Based on DAiSI we implemented the application example as single Dependable Dynamic Adaptive Components. DAiSI binds these Components together at runtime resulting in the Dependable Dynamic Adaptive System described in the application example.

From a Component vendor's point of view, our approach only requires minimal changes to existing Components and makes only few restrictions regarding Component implementations. These few changes – like writing test cases for each used unknown Component– however, result in a higher Dependability of the overall system established at runtime.