ECOOP06 - Poster Submission

Model Level Validation of Component Based Software for Distributed Embedded Systems

Dietmar Schreiner and Karl M. Göschka
Vienna University of Technology
Institute of Information Systems, Distributed Systems Group
Argentinierstrasse 8 / 184-1, A-1040 Vienna
{d.schreiner,k.goeschka}@infosys.tuwien.ac.at

July 2006

1 Summary

When building a component based application for distributed embedded systems, it's overall behavior depends not only on the contracts applying to the components and their interfaces, but even more so on explicit as well as implicit connectors emerging from component composition, deployment and interaction. Explicit connectors provide additional contracts on resource requirements and information channels. We contribute by showing how to perform model level validation of component and contract composition beyond simple interface matching. Moreover, we discuss a classification of typical component connectors to simplify application development for distributed embedded systems. This avoids the need of extensive knowledge of communication subsystems and the existence of any heavy weight middle-ware.

2 Description

Current embedded applications are no longer simple programs executed on single electronic control units (ECUs). In fact, embedded systems applications are nowadays heterogeneous software systems in distributed and very often safety or mission critical environments. This leads to a dramatic increase of software complexity and consequently to an increase of erroneously deployed software. To overcome that problem various paradigms from the classical software engineering process have been adopted to the needs of embedded systems software.

Adoption becomes necessary due to the limited resources in embedded systems, which would otherwise render many useful concepts from the classic software engineering domain unusable. The limitations range from that of processing power over available memory and network-bandwidth up to safety and real-time issues. In general, embedded applications have to be small, efficient and extremely reliable.

A widely accepted and adopted software engineering paradigm within the embedded systems domain is that of component based software engineering (CBSE). The key concept behind CBSE is to construct an application by composing small, simple units of execution - the components. When building a system by connecting components, the point of connection between them, the connector, becomes a hot-spot of abstraction for any interaction. In many component systems like Enterprise Java Beans, the CORBA Component Model, or DCOM, the rather complex process of distributed, heterogeneous interaction is relocated from the individual components into the component model's heavy weight implementation to make it transparent for

the components themselves.

In embedded systems the usage of heavy-weight middleware is often disadvantageous due to the system's limited resources. Nevertheless, it is a good idea to keep the complex and error-prone interaction logic separated, if possible hidden, from the application components. This can be achieved by introducing coherent and explicit connectors and their contracts in a component model. In addition, by using explicit connectors, more precise requirements and provisions regarding the component interconnection become visible. This additional information allows a detailed computation of emerging requirements and may be used for model level validation of component composition.

The poster demonstrates the composition and the deployment of a simple component based application from the automotive domain using UML 2.0 diagrams.

The used diagrams are:

A component diagram: This component diagram shows the application components, their connections and all contracts attached to the components and their interfaces. In conjunction with its transformed version and the deployment diagram, this one forms the center of the poster.

A deployment diagram: The deployment diagram shows the components deployment on two ECUs, connected by one time-driven bus.

A transformed component diagram: This diagram shows the transformed original component diagram, with connectors impersonated by connector-components.

We show, how to (i) make connectors explicit by applying a model transformation. We (ii) identify the generated explicit connectors according to our (iii) classification, which is summarized on the poster as well. By introducing the explicit connectors, additional contracts emerge for the connector and the utilized information channels. Finally we demonstrate the relevance of these connector specific contracts by performing a model level validation of the transformed composition, (iv) calculating composed contracts.