Enhanced Component Interfaces
to Support Dynamic Adaption and Extension

Ralf H. Reussner
Chair Informatics for Engineering and Science
Universität Karlsruhe (TH)
Am Fasanengarten 5, D-76128 Karlsruhe, Germany
reussner@ira.uka.de

Abstract

Current component systems offer the possibility to integrate different enterprise systems, e.g., by wrapping legacy components or integrating several object protocols such as RMI, Corba-IIOP, etc. A disadvantage of today's component systems is that the interface descriptions of their component model do not give alone sufficient information to deploy a component correctly and reliably. Therefore the definition of new interface models, which are enhanced by (semantic) applicability information, play an important role in enterprise application integration. This paper describes a new model of software component interfaces, using an extension of finite state machines to describe the protocol to use a component's offered services, and the sequences of calls to external services the component requires to fulfill its offered services. Our model concentrates on protocol issues of interoperability, such as: (a) Checking whether a component will be used according its protocol in its environment during system integration (i.e., before the component is actually used). (b) Adaptation of the protocol of a component which describes the component's offered services, in case the environment does not offer all resources the component requires to offer all its services in the expected protocol. In this case the adapted component still offers a subset of its services, to the contrary of today's component systems, which do not allow any integration in this case at all. (c) Computation the similarity of two interfaces, e.g., to see which legacy component fits some new requirements best. (d) Extension of component protocols by plug-ins during run-time in a correct, secure and reliable way. In current component systems the interfaces simply do not model enough information for a component to decide whether to accept a plug-in component (because it fulfills all requirements) or to reject it, e.g., due to reasons of security. In our paper we present a description of our new interface model and present the algorithms for integration checking, automatic adaptation and dynamic extension. These algorithms are implemented in our CoCoNutJ Prototype.

1. Introduction

Today's business information systems are confronted with increasing demands for changeability and interoperability. Especially business to business e-commerce requires business information systems to reflect new and fast changing workflows and to interoperate with business systems of other companies [12]. These demands of adaptability and interoperability request for new approaches in software technology, because old monolithic software systems cannot be adapted to changing requirements sufficiently fast. In fact, monolithic systems often show the effect of a fluctuating architecture while being adapted. That is, each unforeseen change slowly destroys the conceptual design and architecture of the system. The result is a even more unmaintainable system, which cannot easily be adapted to new requirements, and often with high costs in time and money.

Component based software development (CBSD) may be useful to create systems of such a high changeability and interoperability. One of the goals of CBSD is to create systems out of reusable and adaptable components; systems are constructed by (re-)configuration of these components. Several definitions for software components are given [22, 6, 2]. Even to increase confusion about components, many claims of CBSD are also made by object or module technology: increased reuse, easier changeability, etc. This gives reason to clarify what we regard as a component in the following and what the differences are of a