Deskriptoren zur Beschreibung der Anwendungsstruktur

Christoph Pohl Steffen Zschaler

24. April 2003

Der Container benötigt verschiedene Deskriptoren, die die Struktur der Anwendung, d.h. das Zusammenspiel der verschiedenen Komponenten(-instanzen) beschreiben. Diese Deskriptoren werden in XML formuliert.

Es wurden die folgenden drei Arten von Deskriptoren definiert:

1. **Komponentenspezifikation** (`*.spec`) Dieser Deskriptor ist angelehnt an CORBA Component Descriptors[1] Er beschreibt die verfügbaren Komponenten und ihre Schnittstellen, trifft aber noch keine Aussagen über Implementierungen oder Instanzen. Insbesondere werden über das Niveau von Java-Interfaces hinaus sog. Component Ports beschrieben, das sind Beschreibungsmöglichkeiten für *provided* und *used* Interfaces, *emitted*, *published* und *consumed* Events sowie *sources* und *sinks* von Strömen.
   Deskriptoren diesen Typs liegen global im Kontext einer Anwendung vor. Sie werden vom Designer oder Programmierer der Komponenten als zusätzliche funktionale Meta-Daten zur Entwurfszeit angelegt.
   Die DTD findet sich in `comquadcomponent.dtd`.

2. **Komponentenimplementationen** (`*.impl`) Dieser Deskriptor beschreibt, welche Implementierungen (*realization*) für eine Spezifikation (*interface*) vorliegen.
   Der Deskriptor wird mit der/den Implementierung(en) in einem Archiv zur Verfügung gestellt, kann aber auch insgesamt mit Schnittstellen, Implementierungen und Assembly einer Anwendung gebündelt werden. Er wird vom Programmierer der Komponente am Ende der Entwicklungszeit erstellt.
   Die DTD findet sich in `componentimpl.dtd`.

3. **Komponentenassembly** (`*.asmbl`) Dieser Deskriptor beschreibt:

[1] OMG CORBA Components Version 3.0, June 2002, formal/02-06-65, http://www.omg.org/cgi-bin/doc?formal/02-06-65

- anwendungsglobal verfügbare Komponenten und deren Instanzen (*homeplacements* und *componentinstantiations* unter *partitioning*),
- Grundgerüste für die Instanzierung von Komponenten zur Laufzeit (*templates*) nebst von diesen implizit erzeugten Instanzen (*implies*) sowie
- die Beziehungen zwischen Komponenten und zwischen deren Instanzen (*connections*). Hier werden die *ports* der Komponenten, wie sie in deren `*.spec` Deskriptoren beschrieben wurden, auf Schnittstellen- und Instanzebene verbunden.

Assembly Deskriptoren können anwendungsglobal definiert, aber auch von einzelnen Komponentenimplementierungen überschrieben werden. Assemblies einzelner (kompositer) Komponenten werden zur Entwicklungszeit vom Programmierer erstellt, wohingegen die Assemblies ganzer Anwendungen von Application Assemblern (Dritte, die Anwendungen aus vorgefertigten Komponenten zusammenstellen) nach der Entwicklung aber vor dem Deployment zusammengefasst werden. Schlussendlich kann der Deployer bzw. Container-Administrator zur Deployment-Zeit noch Anpassungen im Assembly der Anwendung zur konkreten Installation vornehmen.
Die DTD findet sich in `componentassembly.dtd`, ein Beispiel in `bsp.asmbl`.

Es ist wichtig, sich die Unterschiede dieses Konzeptes gegenüber vorhandenen Komponenten-Programmiermodellen vor Augen zu führen: EJB 2.0[2] kennt für alle drei Problembereiche nur einen Deskriptor – `ejb-jar.xml`. CORBA Components definiert zwar einen Component Descriptor zur Erfassung funktionaler Meta-Daten von Komponentenspezifikationen, jedoch existiert kein Deskriptor für Implementierungen. Die Zuordnung von Implementierungen zur Spezifikationen erfolgt implizit im Assembly Deskriptor, wo auch explizit konkrete Implementierungen zur Instanziierung ausgewählt werden. Das Konzept von COMQUAD sieht jedoch vor, dass jeweils die bestgeeigneteste Implementierung aus einem Repository funktional equivalenter (aber mit unterschiedlicher Dienstgüte behafteter) Implementierungen einer Spezifikation vom Container zur Laufzeit ausgewählt wird.

[2]Sun's Enterprise JavaBeans v2.0, http://java.sun.com/products/ejb