Komponenten-Frameworks systematisch entwerfen (Abstract)

Dr. Wolfgang Weck
Unabhängiger Software-Architekt, Zürich
www.wolfgang-weck.ch

Komponenten-Frameworks wurden definiert als eine Menge von Schnittstellen und Interaktions-Regeln, die bestimmen, wie in das Framework ‚eingesteckte' Komponenten zusammenarbeiten. Typische Komponenten-Frameworks enthalten auch ausführbare Programmteile, welche die Einhaltung (eines Teils) der Interaktions-Regeln erzwingen. Die Implementierungen des Komponenten-Frameworks und der erweiternden Komponenten sind und bleiben unabhängige Einheiten.[1]

Dieses Tutorial präsentiert einige Lehren, die sich aus einem einfach gewählten und daher leichtverständlichen Beispiel ziehen lassen: (i) Komposition mit Plug-and-Play braucht im voraus standardisierte Komponenten-Verträge. (ii) Diese Verträge sind meist symmetrisch und bestehen aus mehreren Interfaces (im Sinne gängiger Programmiersprachen), die wiederum verschieden Vertrags-Rollen definieren. (iii) Fehler in Komponenten können für Dritte, welche die Komponenten zu Systemen zusammenbauen, sehr schwer zu klärende Probleme verursachen. Diese Situation lässt sich mit ‚Component Hell' beschreiben. (iv) Beim Entwurf der Komponenten-Verträge können Vorkehrungen getroffen werden, um solche Probleme zu vermeiden bzw. später lösen zu können. (v) Dafür wird es häufig nötig sein, die Verträge mit kleinen ausführbaren Programmteilen anzureichern.

Aus der Reflexion dieser Lehren und des illustrierenden Beispiels lassen sich Regeln für einen systematischen Entwurf von Komponenten-Frameworks ableiten.

Dr. Wolfgang Weck übernimmt Aufgaben zum Architekturentwurf, Schnittstellenmanagement sowie Architekturreviews. Er ist Autor des Kapitels „Die Rolle der Software-Architekten" im „Handbuch der Software-Architektur" (dpunkt-Verlag, 2006). Er unterrichtet an Fachhochschulen in der Schweiz und in Deutschland. Seit der Promotion an der ETH Zürich hat er sich sowohl als Praktiker als auch als Forscher eingehend mit den Konzepten und der Umsetzung von Komponentensoftware und dem dazugehörigen Architekturansatz beschäftigt.

[1] C. Szyperski: *Component Software – Beyond Object-Oriented Programming*, Second Edition. Addison-Wesley, 2002.

© Dr. Wolfgang Weck, 2008