

Vorlesung Software aus Komponenten

1. Komponenten – Markt - Standards

apl. Prof. Dr. Hans-Gert Gräbe
Wintersemester 2009/10

Organisatorisches

- Termin: freitags 13:15 – 14:45 Uhr, Hs 13
- Abschluss: Klausur Anfang Februar
 - Klausur (60 min), Ergebnis anrechenbar als
 - Prüfung im Modul 10-202-2311, wenn Seminarschein vorliegt
 - studienbegleitende Modulprüfung im Kernfach Angewandte Informatik (Diplom-Informatiker)
 - APL (Master-Studiengang)
 - Alle anderen Anrechnungen bedürfen vorheriger Absprache
- Ansprechpartner
 - apl. Prof. Dr. Gräbe <graebe@informatik.uni-leipzig.de>
 - Sprechzeiten: montags 13-14 Uhr oder nach Vereinbarung per Mail
- Informationen:
 - <http://bis.informatik.uni-leipzig.de/HansGertGraebe>
 - <http://olat.informatik.uni-leipzig.de> – BIS-OLAT-Portal

Inhalt dieser Vorlesung

1. Komponenten, Markt, Standards
2. Grundlegende Prinzipien der Wiederverwendung
 - Komponenten, Objekte, Moduln, Schnittstellen
 - Die Vererbungsproblematik
3. Komponentenmodelle und -plattformen
 - Grundlagen
 - OMG und CORBA, Basisdienste
 - Sun und Java, JSP und EJB
 - Microsoft und .Net
 - Neuere Entwicklungen: CCM, Spring, OSGi
4. Komponentenmodelle im Vergleich

Die Vorlesung orientiert sich am Buch [Szyperski, 2002].

Warum Software aus Komponenten?

- „Stehen auf den Schultern von Riesen“
- praktische Wiederverwendung von Softwareteilen
 - kein wiederholtes Schreiben gleichartigen Codes
 - andere Formen: Wiederverwendung von Quellcode, Verwendung von Bibliotheken, Höhere Abstraktionsprinzipien (Design, Architektur)
- Amortisation von Investitionen durch Mehrfachverwendung
- typischer ingenieur-technischer Zugang
 - Vorteile: höhere Qualität, schnellere Entwicklung, flexiblere Reaktion auf wechselnde Anforderungen
- Einsatz wiederverwendbarer Softwareteile ist auch innerhalb eines Unternehmens sinnvoll
 - erfordert konzeptuelle Vorbereitung und Standardisierung
- Softwarekomponenten und Markt

Software-Erstellung als ingenieurtechnischer Prozess

Standardsoftware

hohe Stückzahl,
große Einsatzbreite

Parallelen zum
Werkzeugmaschinenbau

Auftragssoftware

geringe Stückzahl,
spezielle Einsatzbedingungen

Parallelen zum Anlagenbau

Software als Produkt

Komponenteneinsatz innerhalb
eines Unternehmens im
Rahmen von Produktlinien

Software als Prozess

Komponenten als unternehmens-
übergreifende Infrastruktur:
Bausteine, aus denen Applikationen
gefertigt sind

Was sind Software-Komponenten?

- „Components are for composition“
 - Komposition erlaubt es, vorfabrizierte „Dinge“ wiederzuverwenden, indem sie in immer neuen Kombinationen zusammengesetzt werden.
 - Allgemeiner Wiederverwendungsansatz ist für unsere Betrachtungen zu breit
 - Ansatz muss den Aspekt „Aufbau aus Teilen“ stärker berücksichtigen
 - es reicht nicht, monolithische Software für Wiederverwendung zu „zerstückeln“
 - Wiederverwendungsansatz muss bereits bei der Erstellung der Komponenten berücksichtigt werden, nicht erst a posteriori
 - Funktionalität muss genügend allgemein geplant sein, um Wiederverwendung in ausreichender Anzahl verschiedener Kontexte zu ermöglichen
 - Verallgemeinerung muss genügend speziell sein, um Wiederverwendung praktisch sinnvoll zu ermöglichen
 - Wiederverwendung soll über Firmengrenzen hinaus möglich sein

1.1. Einführung

Eine erste Komponenten-Definition

- Definition des Begriffs „Software-Komponente“ ist auf unterschiedlichen Abstraktionsebenen möglich

Software-Komponenten sind ausführbare Software-Einheiten, die unabhängig hergestellt, erworben und konfiguriert werden und aus denen sich funktionierende Gesamtsysteme zusammensetzen lassen. [Szyperski]

- Zusammensetzungsaspekt steht im Vordergrund
 - dafür sind Standards erforderlich:
 - Einordnung in ein spezielles Komponentenmodell
 - ausführbar auf einer speziellen Komponentenplattform
- So zusammengesetzte Systeme heißen Komponenten-Software
- Unabhängigkeit und Ausführbarkeit
 - Unabhängigkeit von Entwicklung und Herstellung
 - Abstraktionen auf Quellcode-Ebene ausgeschlossen
 - älteste Beispiele für Software-Komponenten: Bibliotheken

Vorteile des Komponentenansatzes

- Die Software-Krise großer monolithischer Anwendungen
- in-House-Lösung: Modularisierung und objektorientierte Ansätze
 - Problem: Wiederverwendung nur eigener Expertise
 - für komplexe Produkte muss die Expertise auf der ganzen Front vorgehalten werden
 - maximal von großen Firmen (SAP, Microsoft, etc.) zu schultern
 - nur für Software-Produkte sinnvoll einsetzbar
- Beispiel Open-Source-Projekte
 - Vorteil ist weniger die Quelloffenheit als die Verfügbarkeit von qualitativ hochwertigen Software-Bausteinen
 - typisches Ergebnis arbeitsteiligen Vorgehens wie im Wissenschaftsbetrieb üblich
- Beispiel klassische Ingenieurtechnik: Alle entwickelten Ingenieurdisziplinen verwenden Komponententechnologien
- Komponenten-Software ist unter diesem Blickwinkel **der** Ausweg aus der Software-Krise