

Vorlesung Software aus Komponenten

1. Komponenten – Markt – Standards

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2015/16

Organisatorisches

- Termin: montags 9:15 – 10:45 Uhr, Hs 19
- Abschluss: Mdl. Prüfung, Ergebnis anrechenbar als Prüfung im Modul 10-202-2311, wenn Seminarschein vorliegt
 - Alle anderen Anrechnungen bedürfen vorheriger Absprache
- Ansprechpartner
 - Prof. Dr. Gräbe <graebe@informatik.uni-leipzig.de>
 - Sprechzeiten: nach Vereinbarung per Mail
- Informationen:
 - <http://bis.informatik.uni-leipzig.de/HansGertGraebe>
 - <https://olat.informatik.uni-leipzig.de> – BIS-OLAT-Portal, Kurs W15.BIS.SWK

Relevante Seminare und Praktika

- Seminar Anwendungen Semantischer Technologien (Arndt, Elze Gräbe, Martin),
 - Montags 11-13 Uhr, Raum P-801, nicht wöchentlich
- Seminar „Data as a Component“ (Hellmann, Kühne)
 - Freitags 9-11 Uhr, Raum P-801, nicht wöchentlich
- Einschreibung in die Seminare im Kurs W15.BIS.Seminare
- Praktikum im Kurs „Gesellschaftliche Aspekte des digitalen Wandels“ (Gräbe), n.V.
 - Einschreibung W15.BIS.Wahl bis 18.10.

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

Software als Produkt

Komponenteneinsatz inner-
halb eines Unternehmens im
Rahmen von Produktlinien

Auftragssoftware

geringe Stückzahl,
spezielle Einsatzbedingungen

Parallelen zum Anlagenbau

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, vorgefertigte „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 Verwendung
 - Abstraktionen auf Quellcode-Ebene sind damit 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

Komponentenmarkt

Ansatz brummt richtig, wenn die erforderlichen Komponenten in guter Qualität und ausreichend breiter Funktionalität verfügbar sind

- Modell Komponentenmarkt vs. Open Source und GPL

Komponentenmärkte existierten bisher immer nur in Ansätzen

- Warum Mehrzahl?
- technologische und ökonomische Rahmenbedingungen erforderlich
- technologische Bedingungen existieren seit Ende der 60er Jahre
- Markt erfordert Standard, da Komponenten nur in einer entsprechenden Infrastruktur (Komponentenplattform) operieren können.
- Standardsetzung ist ein *politisches* Problem

Warum ist Komponenten-Software auch in Prä-Marktpphase attraktiv?

- Vorbereitung auf kommenden Markt
- Übergang zu komponentenbasierter Software ist aufwändig
- Setzen von Standards und Sammeln von Kompetenz

Die selbstverstärkende Wirkung eines Komponenten-Markts

Wird es wirklich dauerhaft Komponenten-Märkte geben?

- Bindung an Standards einer gewissen IT-Generation
- Infrastruktureller Charakter einer Software-Komponenten-Szene
- Ökonomische Besonderheiten von Software und deren inhärente Tendenz zur Monopolisierung