Vorlesung
Software aus Komponenten

2. Grundlagen

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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten und Entwurf

K. als Einheit der Abstraktion

- Ansatz: white box ➔ black box
- Entwurfsexpertise kapseln (design expertise ready for use)
- Theoretische Einschränkung der Vielfalt in der aktuellen Ebene ist Basis für größere Vielfalt in der nächsthöheren Ebene

K. als Einheit der Analyse

- Analyse an vielen Stellen im Komponentenlebenszyklus erforderlich (Spezifikationsprüfung, Tests, Re-Engineering)
- Kopplung zwischen Einheiten bestimmt, wie weit eine individuelle Analyse zweckmäßig bzw. überhaupt möglich ist
- Regel: Einheiten für Analyse so klein wie möglich
- Regel: streng statische hierarchische Grenzen (Moduln, Subsysteme) erleichtern die Analyse

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten aus technischer Anbietersicht

K. als unabhängig compilierbare Einheit

- Compilierbarkeit und Analyse sind eng verbunden
  - white box: Analyse
  - black box: Compilierbarkeit
- sinnvolle Einheiten: Klassen oder Pakete?
- Globale Optimierung?
  - Antwort 1: Einheiten möglichst groß wählen
  - Antwort 2: Optimierung zwischen Komponenten, vielleicht sogar erst nach der Lokalisierung
- muss im Komponentenkonzept und in der Komponentenbeschreibung verankert sein

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

K. als Packungs-Einheit

- Entpackung (deployment) = Prozess der Vorbereitung der Komponente auf den Einsatz in einer speziellen Umgebung (Lokalisierung)
  - wurde lange nicht als separater Schritt betrachtet
  - Prozess der Anbindung an eine spezielle Komponentenplattform
- Konfiguration = Einstellung spezieller Eigenschaften der Komponente für den konkreten Einsatz
- Installation = plattformspezifische Aktivität, mit der eine entpackte Komponente für die Nutzung in einer speziellen Hardware-Konfiguration verfügbar gemacht wird, die von der Plattform unterstützt wird.
  - Zeit, in der auch kritische Tests ausgeführt werden, die vor dem eigentlichen Betrieb erfolgen müssen (etwa Integritätstests)
- für alle drei Aktivitäten müssen entsprechende Beschreibungen erstellt werden

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten aus technischer Nutzersicht

K. als Einheit des Ladens

- oft wird beim ersten Einsatz einer Funktionen erst entsprechender Code geladen.
- benötigt Mechanismen des dynamischen Ladens (etwa DLL)
- typisch werden dabei gleich mehrere zusammenhängende Klassen geladen
  - Komponente als Einheit des Ladens sinnvoll
- Kontrolle der Version
  - Maximale Flexibilität, wenn die Versionskontrolle auf der Ebene von Schnittstellen oder sogar Methoden erfolgt
  - Erfordert eine Zuordnung von Methodenversionen zu Komponentenversionen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

K. als Einheit des Ladens (Fortsetzung)

- Java: Versions-Check erst zur Laufzeit
  - ist problematisch, da Inkompatibilitäten erst mitten in der Programmausführung entdeckt werden
  - Kontrolle der Erfüllbarkeit der Importrelationen
- Kontrolle von Namenskollisionen
  - Lösung: Hierarchisches Namensraum-Konzept
  - Problem der Namenskollision: Versionen derselben Komponente
- Konsistenz bereits geladener Komponenten muss erhalten bleiben
  - Konsistenz muss dazu lokale Eigenschaft sein
  - schließt z.B. globale Typprüfungsansätze aus

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

K. als Einheit der Lokalität

- Problem im Kontext verteilter Systeme: was befindet sich (lokal) auf welchem Rechner
  - typisch sind hierarchische Konzepte des Clusterns von Rechnern
  - SAN, LAN, WAN, Internet (und weitere Zwischenstufen)
  - unterschiedliche Kommunikationszeiten und -kosten
- Tradeoff: minimale Kommunikationskosten vs. maximale Ressourcennutzung
- hohe Kopplung zwischen Objekten aus derselben Komponente
  - Komponente sollte nicht auf verschiedene Prozesse oder Maschinen verteilt sein
  - Bündelung von Zugriffen auf Objekte über Prozessgrenzen hinweg
  - ein komplexer statt mehrerer einfacher Zugriffe

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten und Management

K. als Einheit der (Auseinandersetzung um) Fehlersuche

- Problem: Was ist (u.a. wer haftet?), wenn ein aus Komponenten zusammengebautes Produktivsystem fehlerhaft arbeitet?
- Problem der Lokalisierung von Fehlern (und damit Verantwortlichkeiten)
  - besonders schwierig wird es, wenn Objektreferenzen die Komponentengrenzen verlassen
- vitale Regel: Fehler müssen in den verursachenden Komponenten bleiben (bug containment)
  - typische nicht-lokalisierbare Fehler: Speicherzugriffsfehler
- Folge: Ausnahmebehandlungen müssen in der Regel innerhalb einer Komponente bleiben
  - Ausnahmen davon sind im Komponenten-Kontrakt zu fixieren
  - Komponente als Einheit der Fehlerbehandlung

K. als Auslieferungseinheit

- Bündel der technischen und wirtschaftlichen Aspekte
- Management (Service, Wartung, Schulung, Updates, ...) treibt den Preis in die Höhe
- betriebswirtschaftliche Bedeutung jenseits der (geringen) Replikationskosten

K. als Einheit der Kostenrechnung

- wichtig in größeren industriellen Kontexten, um Projektentwicklungskosten verfolgen zu können

K. als Einheit des Managements

- oft zu klein, Management auf der Ebene von Subsystemen, die mehrere Komponenten zusammenfassen
- etwa auf der Ebene der Server

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten-Modelle

1. Grundlagen: Kommunikationskonzepte
2. Erste Komponentenansätze
3. OMG und CORBA, Common Object Request Broker Architecture
4. Sun und Java, Servlets und JSP, EJB
5. Microsofts und .NET, Common Language Runtime
6. Neuere Komponenten-Frameworks

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Interprozess-Kommunikation (IPC) auf OS-Ebene

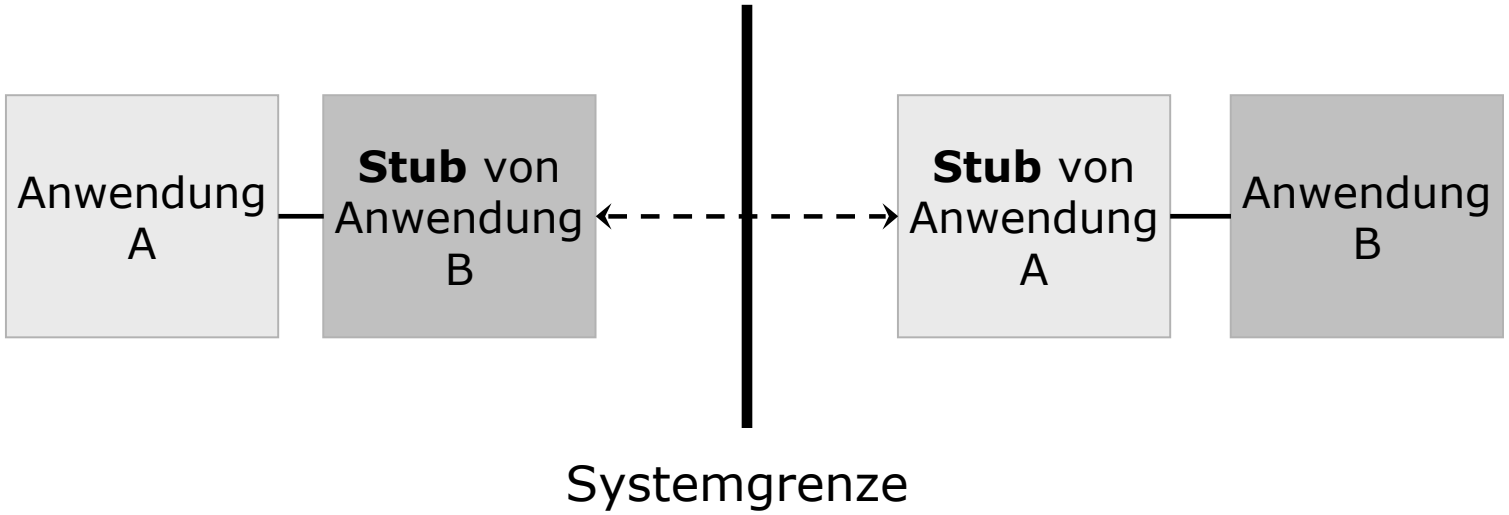
- Charakteristika
  - kaum plattformübergreifend standardisiert
  - nicht Teil des von-Neumann-Modells
  - Prozess = virtueller Rechner auf physischem Host
- IPC-Modelle: Dateien, Pipes, Sockets, Semaphore, shared memory
  - außer Sockets keins so weit standardisiert, dass es plattformübergreifend eingesetzt werden könnte.
  - außer shared memory skalierbar und internetfähig
  - operiert auf Bitebene -> zu kompliziert für komplexe Anwendungen

IPC operiert auf Bitebene und ist deutlich zu kompliziert für komplexe Anwendungen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Remote Procedure Calls (RPC, 1984)

- Ansatz: Stubs, die auch entfernte Prozeduraufrufe lokal aussehen lassen
  - Aufgabe des Stub: Serialisierung bzw. Deserialisierung des Prozeduraufrufs und der Aufrufparameter unter Beachtung von plattformabhängiger Byte-Kodierung, Zahldarstellung, ...



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Nutzung leichtgewichtiger RPC zur IPC auf derselben Maschine (Windows NT)
- Vorteil: einheitliches Abstraktionsniveau für alle Kommunikationserfordernisse (innerhalb eines Prozesses, zwischen Prozessen, zwischen Computern)
- Nachteile:
  - Versteckte Kommunikationskosten (Unterschied um Faktor $10...10^4$), Client kann nicht unterscheiden, ob lokaler oder entfernter Aufruf
  - blockierendes Konzept
  - Umgang mit Versionierung und Evolution von Komponenten vollkommen unklar

Das RPC-Konzept bildet zusammen mit dynamisch linkbaren Bibliotheken (DLL) die Basis für das einfachste Komponenten-Framework (und ist das heute am weitesten verbreitete).

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Objekte und Methoden

- RPC ist ein statisches Aufrufkonzept
- Besonderheit von Methoden gegenüber Prozeduren:
  - werden dynamisch an Hand der Charakteristika des Objekts (=Instanz seiner Klasse) ausgewählt
    - erst nach dieser Auswahl kann der RPC-Mechanismus greifen
    - Klassen müssen dazu genügend (binär kodierte) Information bieten, die durch Introspektion zur Laufzeit abgefragt werden kann
  - Objektreferenzen als Aufrufparameter
    - Keine automatischen Objektkopien

Der RPC-Ansatz ist deutlich aufzubohren, wenn mit Objekten und Methoden mit laufzeitabhängigem Verhalten umgegangen werden soll.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Erweiterung des RPC-Konzepts

- Verwendung von Funktionsvariablen, die zur Laufzeit mit einem Funktionszeiger als Wert belegt werden
- Virtuelle Methodentabellen (VMT)
  - Beispiel: COM‘s dispatch tables (Microsoft)

oder Spezielle Laufzeitumgebung (virtual machine, VM)

- übernimmt Management von *Methoden*-Aufrufen
  - SOM (IBMs System Object Model)
  - Java (Java virtual machine)
  - .NET common language runtime (CLR)
- braucht spezielle Mechanismen, um **über die Grenzen der VM hinaus** mit Komponenten zu kommunizieren

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Über RPC-Mechanismus hinaus gehende Fragen, die ein objektorientiertes Kommunikationskonzept beantworten muss

1. Wie werden Schnittstellen spezifiziert?
2. Wie werden Objektreferenzen behandelt, wenn der lokale Bereich verlassen wird?
3. Wie werden Dienste aufgefunden und bereitgestellt?
4. Wie wird die Evolution von Komponenten gehandhabt? (Versionsmanagement)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Schnittstellenspezifikation

- **Definition:** Interface ist ein abstrakter Datentyp
  - Sammlung von Operationsbezeichnern mit ihren Signaturen
  - Signatur = Typ und Aufrufmodi der Parameter

- **Schnittstellen-Beschreibung** durch IDL
  - mehrere Standards koexistieren (insb. OMG IDL und COM IDL)
  - Java und CLR: Keine IDL, sondern sprachspezifisches Meta-Datenformat, das auf jede der IDL abgebildet werden kann
    - dazu sind entsprechende Abbildungen zu spezifizieren
      - *Java to IDL language mapping specification* (Version 1.3 vom Sept. 2003, siehe http://www.omg.org)
    - **Umgekehrt:** Java-Werkzeug **idlj** erzeugt aus einer (OMG) IDL-Beschreibung (u.a.) ein Java *interface*.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Namensgebung und Auffinden von Diensten

- Dienste werden über ihren Namen identifiziert
  - OMG: UUID als Standard der Open Software Foundation (DCE)
    - genügend lange Zeichenkombinationen
  - COM (Microsoft) verwendet modifizierte Version: Global Unique IDentifier (GUID)
    - Namensgebung für Interfaces (IID), Gruppen von Interfaces (categories = CATID) und Klassen (CLSID)
    - CLR: Identität durch private / public-key auf Komponentenebene
  - Java: Eindeutigkeit über zusammengesetzte Pfadnamen (Anlehnung an URL)

- Über den Namen muss wenigstens folgende Funktionalität zur Laufzeit abrufbar sein:
  - Typtest der Schnittstellen
  - Introspektion der Schnittstellen
  - dynamisches Erzeugen neuer Objekte

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Der dokumentenzentrierte Ansatz

- Idee: Nutzer wird nicht mit vielen verschiedenen Applikationen konfrontiert, sondern mit Dokumenten, die aus mehreren Teilen bestehen können. Diese Teile können unterschiedliche Applikationen zur Darstellung benötigen, kennen diese aber selbst.

- Erste Realisierung unmittelbar auf der Ebene von integrierten Textdokumenten
  - Hypercard (Apple)
  - Word mit Visual Basic und VBX (Microsoft)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Visual Basic

- Dokument besteht aus (mehreren) Formularen
- Formular kann mit Kontrolleinheit ausgestattet sein
- Kontrolleinheiten interagieren über Basic-Skripte

Flexibilität und Produktivität dieses Konzepts führten zur Herausbildung des ersten Komponentenmarkts mit Komponenten etwa zur Tabellenkalkulation oder zur Prozessautomatisierung.

OLE als Weiterentwicklung dieses Ansatzes

- Formulare ⇒ Container für beliebige Anwendungen
- Kontrolleinheit ⇒ Dokumentenserver
- Container können hierarchisch ineinander geschachtelt werden

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Der webzentrierte Ansatz

- Idee: Einbettung von beliebigen Objekten in HTML-Seiten
  - z.B. Java Applets, Form-Bestandteile
- Einheitliche und erweiterbare Darstellung im Browser durch Plugin-Technologie
- Schritt weg vom OLE-Containerkonzept und zurück zum (nicht hierarchischen) Formularansatz von Visual Basic

Aktuelle Entwicklungsrichtungen

- COM (Microsoft)
- CORBA (Object Management Group)
- Java (Sun und inzwischen auch IBM)
- Webservices als lose gekoppeltes Konzept