Vorlesung
Software aus Komponenten

1. Komponenten – Markt - Standards

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2005/06

Organisatorisches

- Termin: freitags 9:15 – 10:45 Uhr, KH 2-11
- Abschluss: Klausur Anfang Februar
  - genauer Termin wird noch festgelegt
  - Ergebnis anrechenbar als
    - studienbegleitende Modulprüfung im Kernfach Angewandte Informatik (Diplom-Informatiker)
    - APL (Master-Studiengang)
    - PL bedarf gesonderter rechtzeitiger Absprache
- Ansprechpartner
  - Prof. Dr. Gräbe <graebe@informatik.uni-leipzig.de>
  - Sprechzeiten: montags 13-14 Uhr oder nach Vereinbarung per Mail
- Informationen: Über meine Webseite

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
   - Sun und Java
   - Microsoft und .Net
   - Vergleich
4. Komponenten und Architekturen
5. Komponenten und IT-Professionals

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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, präfabrizierte „Dinge" wiederzu-verwenden, 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

Eine erste Komponenten-Definition

- Definition des Begriffs „Software-Komponente“ ist auf unterschiedlichen Abstraktionsebene 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
- Vorteil von Open-Source-Projekten ist weniger die Quelloffenheit als die Verfügbarkeit von qualitativ hochwertigen Software-Bausteinen
  - typisches Ergebnis arbeitsteiligen Vorgehens wie im Wissenschaftsbetrieb üblich
  - Alle entwickelten Ingenieurdisziplinen verwenden Komponententechnologien
- Komponenten-Software ist unter diesem Blickwinkel **der** Ausweg aus der Software-Krise

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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 existieren erst 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
  - Beispiel: Markt für VBX

Komponentenmarkt

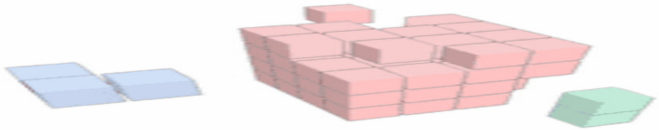
- Warum ist Komponenten-Software auch in Prä-Marktphase attraktiv?
  - Vorbereitung auf kommenden Markt
  - Übergang zu komponentenbasierter Software ist aufwändig
  - Setzen von Standards und Sammeln von Kompetenz

- Die selbstverstärkende Wirkung eines Komponenten-Markts

- Wird es wirklich dauerhaft Komponenten-Märkte geben?
  - Der infrastrukturelle Charakter einer Software-Komponenten-Szene
  - Ökonomische Besonderheiten von Software und deren inhärente Tendenz zur Monopolisierung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- 4 Haupteigenschaften von Komponenten:



… eine funktional und technisch abgeschlossene ausführbare Einheit

… unabhängig als Einheit entwickel- und konfigurierbar

Eine Komponente ist…

… wiederverwendbar

… nur durch genau spezifizierte Schnittstellen ansprechbar

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Baustein-Charakter
  - „Alle Welt ist aus Bausteinen, nur die Software-Branche nutzt dieses Konzept noch nicht“
  - Problem: Bauplan- und Metaprodukt-Charakter von Software
    - Unterscheidung zwischen Software und deren Instanzen
    - Unterscheidung zwischen Blaupausen und Produkten
- Komponenten als Einheiten der Packung (deployment)
    - beyond object oriented programming [Szyperski 02]
    - OO has failed but component software is succeeding [Udell 94]
- Erfolgsgeschichten von Komponenten-Software
  - die älteste: moderne Betriebssysteme
  - Datenbanken und Transaktionsmonitore
  - Plugin-Architektur (nicht nur) moderner Browser
  - moderne Applikationsserver

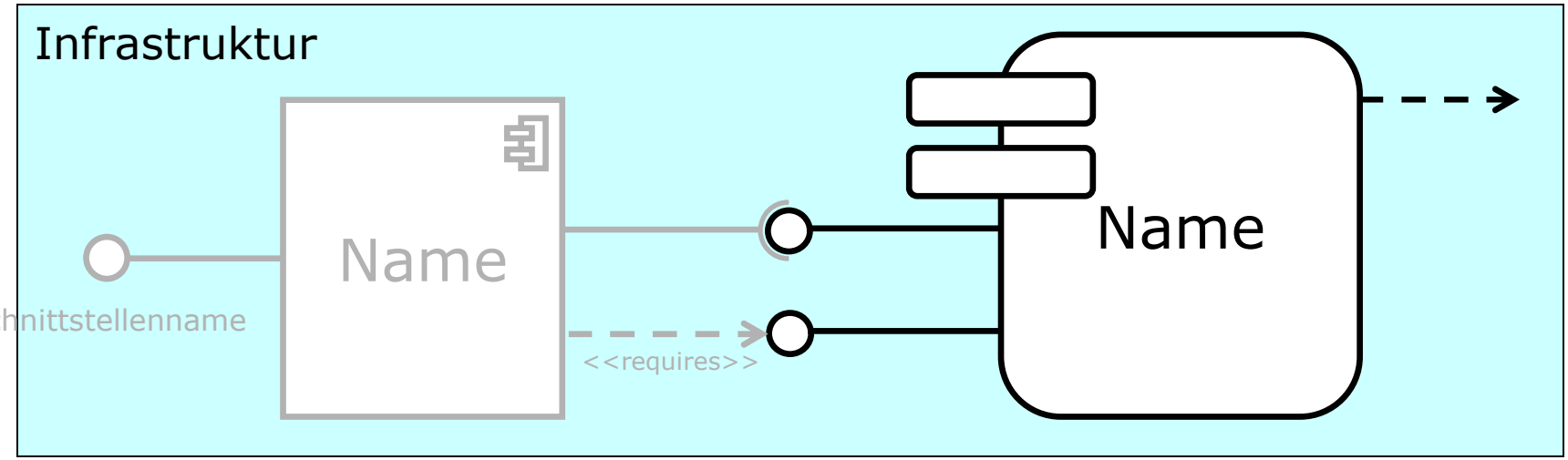
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Gemeinsamkeiten der Erfolgsgeschichten

- Existenz einer **Infrastruktur**
  - grundlegende Funktionalität für Interoperabilität wird in ausreichendem Umfang zur Verfügung gestellt
- Komponenten haben **genügend substanzielle Funktionalität**, die eine wiederholte Entwicklung uneffektiv macht
- Komponenten unabhängiger Anbieter können in der Infrastruktur koexistieren
  - Zusammensetzbarkeit ist eher wahrscheinlich als garantiert
  - plug and play
- Komponenten existieren auf einer Abstraktionsebene, die eine **direkte Bedeutung** für den verteilenden Client haben
    - Bsp. VB: Control hat direkte grafische Repräsentation
  - im Gegensatz zu Objekten, die für Nichtprogrammierer keine eigenständige Bedeutung haben
    - aber: Objekttechnologie als der beste Weg zur Realisierung von Komponententechnologie

Aufbau einer Komponente

- Schnittstellen
  - Operationen / Funktionalität
  - Attribute / Signatur
  - Veröffentlichte Schnittstellen
  - Konsumierte Schnittstellen



UML 2

UML 1.4

Komponentenrelevante Konzepte

Komponenten

- **Gekapselte**, generalisierte **Softwareobjekte**, die einen **Dienst** zur Verfügung stellen und aus denen größere Komponenten oder Systeme gebaut werden können

Kapselung

- wohlspezifizierte Dienste-Schnittstelle, Kontextunabhängigkeit

Generalisierung

- Parametrisierung, Erweiterbarkeit, Nutzbarkeit in unterschiedlichen Anwendungen

Dienst

- Zusammenhängende Sammlung von in Beziehung stehender Funktionalität. Nachfrage nach Dienst (service) von Klienten (clients)

Systemfähigkeit

- Kaskadierbarkeit, Katalogisierbarkeit, Aufbau nach vorgegebenen Architekturprinzipien

Komponenten-Lebenszyklus

- unterscheide zwischen
  - Komponente als Konzept
  - Komponente als auslieferbare prototypische Einheit
  - Komponente als in einem prototypischen Systemkontext konfigurierbare Einheit
  - Komponente als in einem konkreten Systemkontext zu verteilende, zu konfigurierende und zu installierende Einheit
    - unterscheide „deployed" und „installed"
  - Komponenten-Instanz als Instanz einer installierten Komponente

- Komponenten und Dienste: der Begriff „Dienst" wird oft auch im Sinne einer Verbindung mit einem auf einem Markt positionierten Dienstanbieter verwendet
  - Dienste in diesem Sinne sind orthogonal zum Komponenten-konzept und können Komponenten-Instanzen verwenden
  - dazu gehört aber eine konkrete Hardware-, Software- und Organisations-Infrastruktur

Infrastruktur für Komponenten

- Komponenten-Plattform
  - Hilfsmittel und Technologien, die zur Erstellung und zum Betrieb bzw. zur Anpassung flexibler und erweiterbarer Anwendungen auf der Basis von Komponenten erforderlich sind

- Komponenten-Entwicklung
  - Modelle, Methoden und Werkzeuge, die zur Analyse, Entwicklung und Design von auf Komponentensoftware beruhenden betrieblichen Anwendungssystemen dienen
  - Design to / from / for component

- Design **for** Component
  - Initiale Entwicklung atomarer Komponenten zum Ziele der Bereitstellung spezifischer, gekapselter Dienst, welche später in neue Anwendungen schnell und einfach integriert werden können.

- Design **from** Component
  - Inkrementelle Entwicklung von komplexeren Komponenten und Anwendungssystemen unter Nutzung vorhandener und noch zu erstellender Bausteine, sowie der Dienste der Komponenten-Plattform.

- Design **to** Component
  - Methoden zur Transformation konventionell erstellter Anwendungssysteme in eine flexible komponentenbasierte Umgebung.

- Fokus: Komponente als Endprodukt
- Anbieter / Produzenten-Sicht
- Ausrichtung an:
  - Standards
  - angestrebte Zielumgebung
- Ergebnis:
  - Bereitstellung von Komponenten
  - Erfüllung einer spezifischen Aufgabe

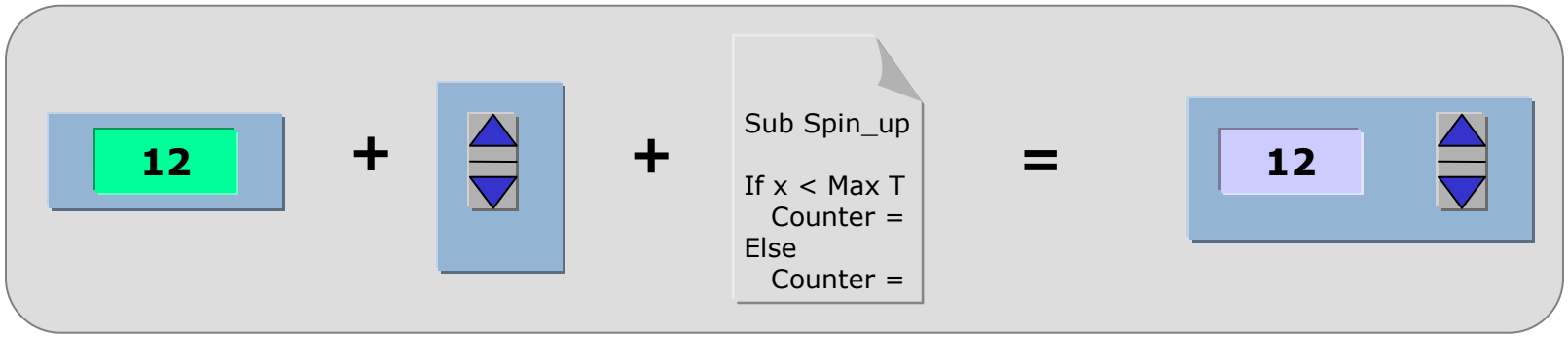


Abb: Zusammenbau einer Komponente aus Teilkomponenten

- Fokus: Zusammenbau komplexer Anwendungssysteme
- Käufer / Anwender-Sicht
- Aggregation von Komponenten auf höheren Niveau
- Grundlagen:
  - Spezialisierte Komponenten
  - "Komponenten-Kleber"
  - Dienste der Komponenten-Plattform
- Ergebnis:
  - Aufgabenspezifische Anwendungskomposition

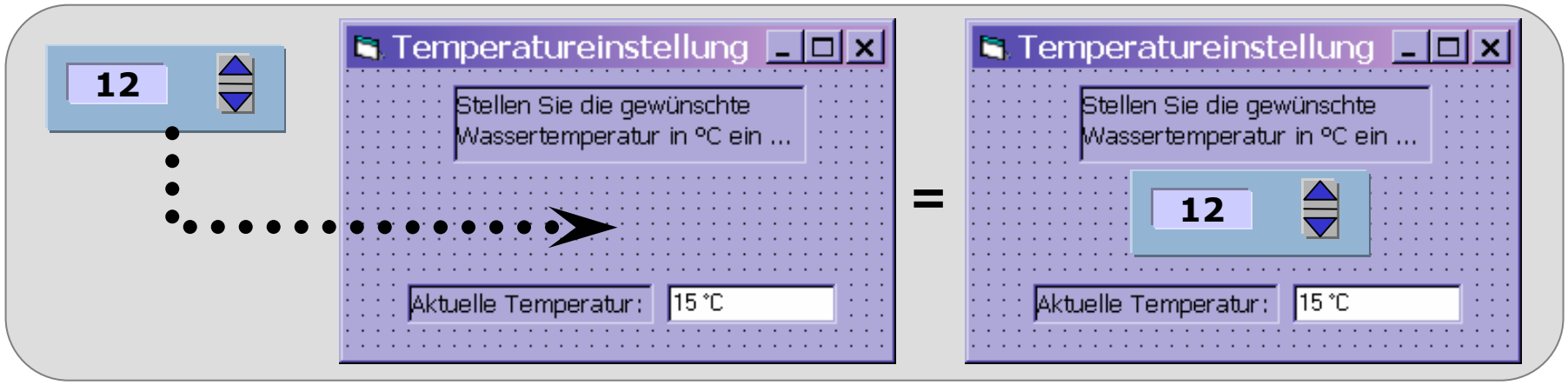


Abb: Verwenden einer Komponente

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Migration zu komponentenbasierten Lösungen
- Anwendungs-Reengineering
  - "Zerschlagen" von Altanwendungen
  - Monolithische Altanwendung wird zu einer flexiblen neuen komponentenbasierten Anwendung umgebaut
- Verwendung vorhandener Komponenten
- Ersetzung alter Komponenten durch neue Komponenten

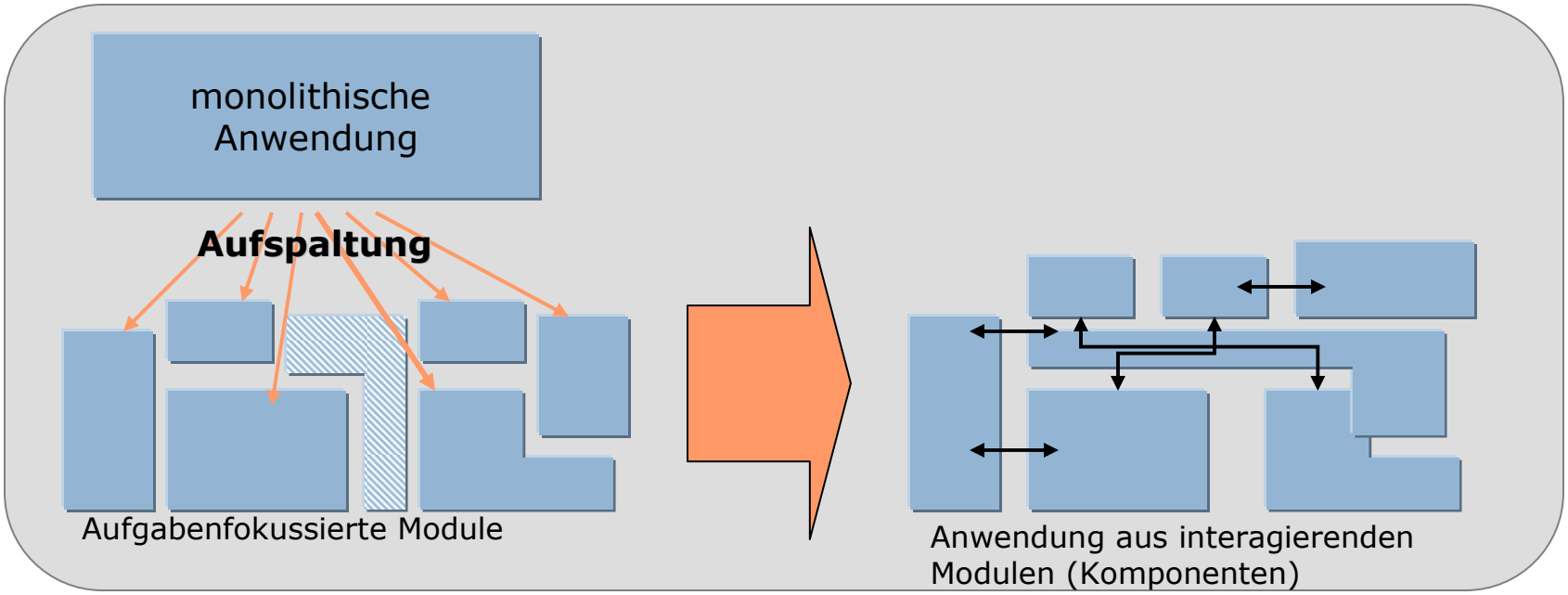
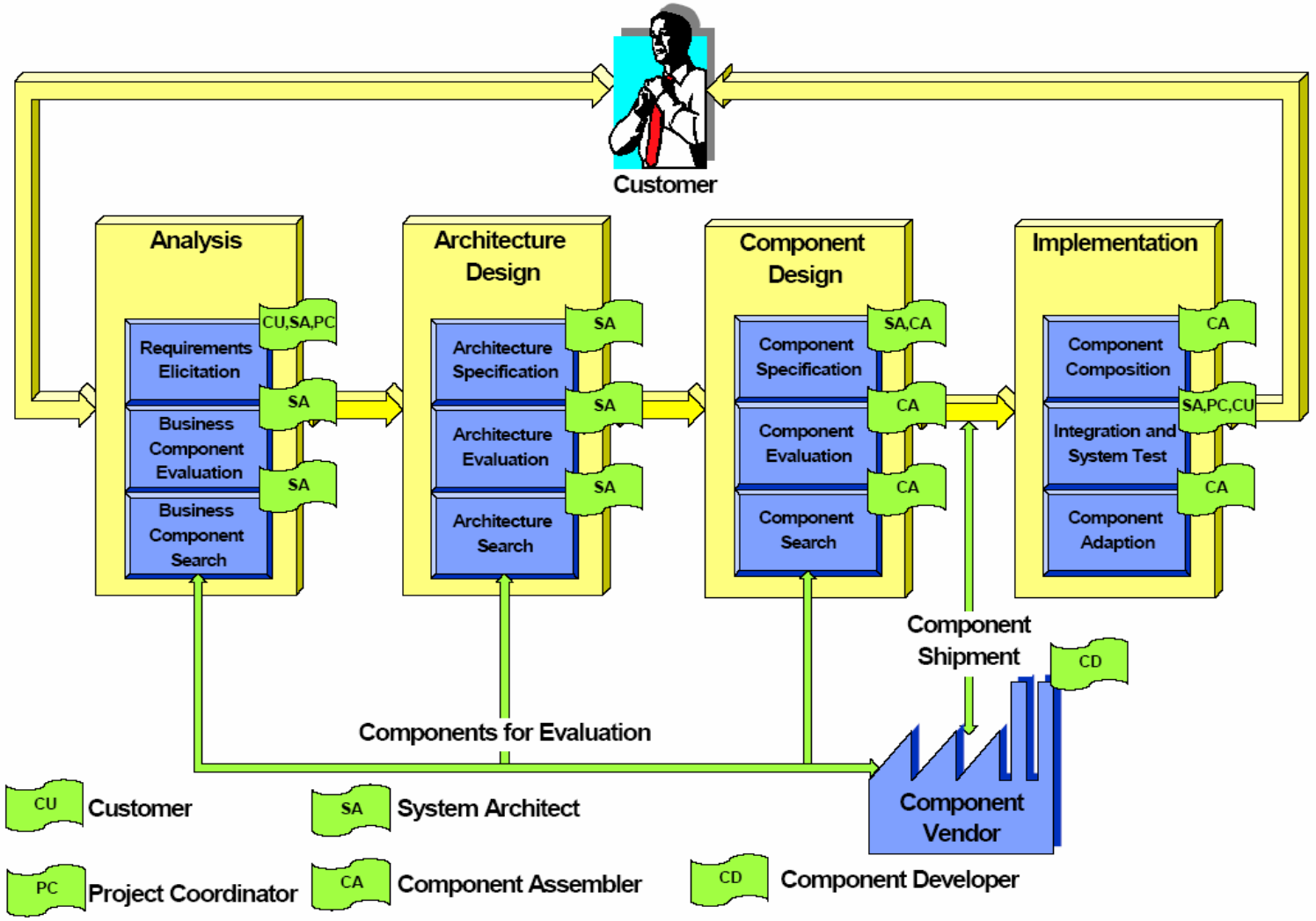


Abb: von einer monolithischen zu einer modularen Anwendung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Entwicklungszyklus für Komponenten-Software



Quelle: /Sihling 1998/

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- ***Komponententechnologie:***
  Mehrstufiger modularer Softwareaufbau aus vorgefertigten Komponenten, in denen Funktionalität so gekapselt ist, dass diese auf einheitliche Weise wiederverwendet werden kann

- ***Anforderungen:***
  - formal fundiertes **Komponentenkonzept** als Basis
  - **Beschreibungstechniken** für derartige Komponenten
  - Entwicklung eines **Prozessmodells** zur Entwicklung, Verwaltung und Zusammensetzung von Komponenten
    - Unterstützung der Zuweisung verschiedener Rollen
  - **Werkzeuge**, welche die Beschreibung und das Prozessmodell unterstützen
    - zur Systemgenerierung selbst
    - zur Dokumentation
    - zur Verifikation und Sicherung wichtiger und kritischer Systemeigenschaften