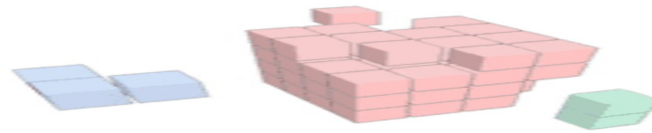


Vorlesung Software aus Komponenten

1. Komponenten – Markt – Standards

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2010/11

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

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

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

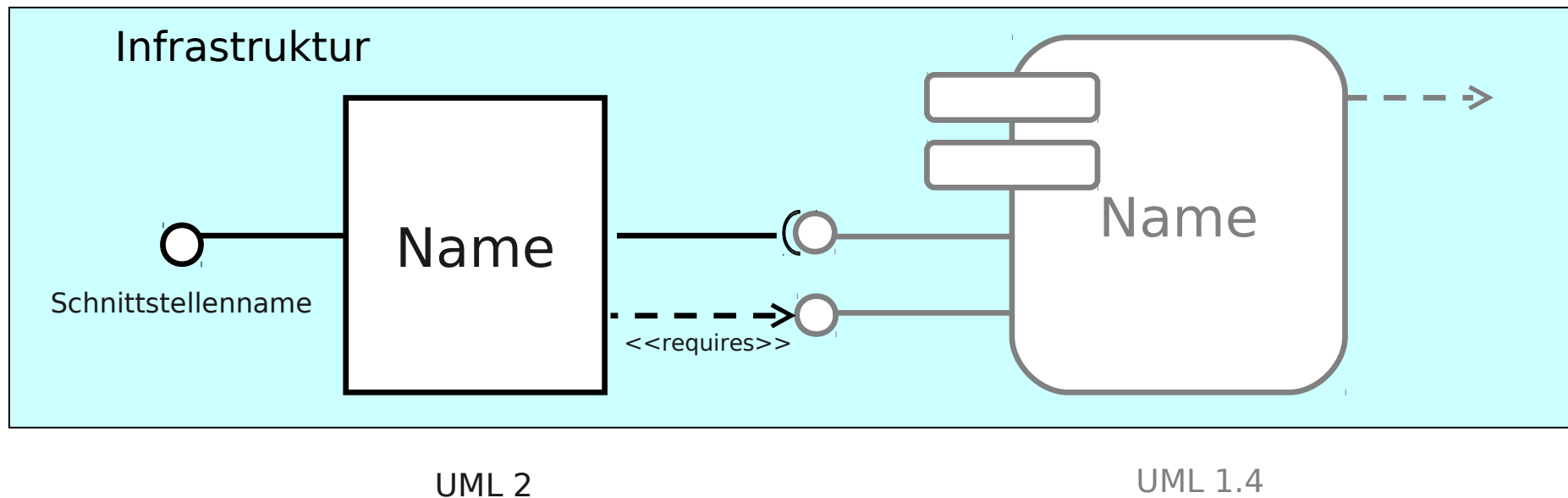
Komponenten existieren auf einer Abstraktionsebene, die eine **direkte Bedeutung** für den verteilenden Client haben

- 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

1.3. Komponenten

Aufbau einer Komponente

- Schnittstellen
 - Operationen / Funktionalität (funktionale Kopplung)
 - Attribute / Signatur (Datenkopplung)
 - Veröffentlichte Schnittstellen (exportierte Dienste)
 - Konsumierte Schnittstellen (importierte Dienste)



1.3. Komponenten

Komponentenrelevante Konzepte

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

1.3. Komponenten Infrastruktur für Komponenten

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

1.4. Komponentenentwicklung

Designprinzipien

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.

1.4. Komponentenentwicklung

Design for Component

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

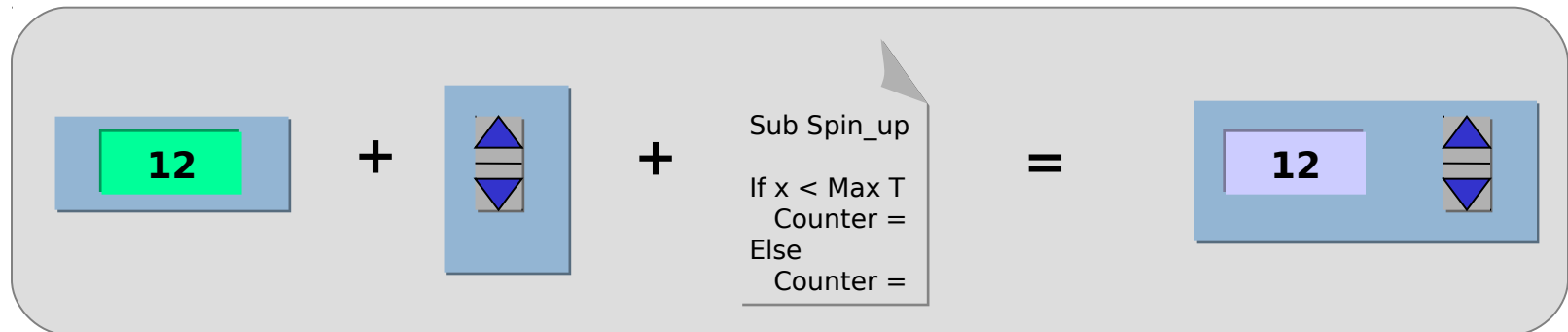


Abb: Zusammenbau einer Komponente aus Teilkomponenten

1.4. Komponentenentwicklung

Design from Component

- 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

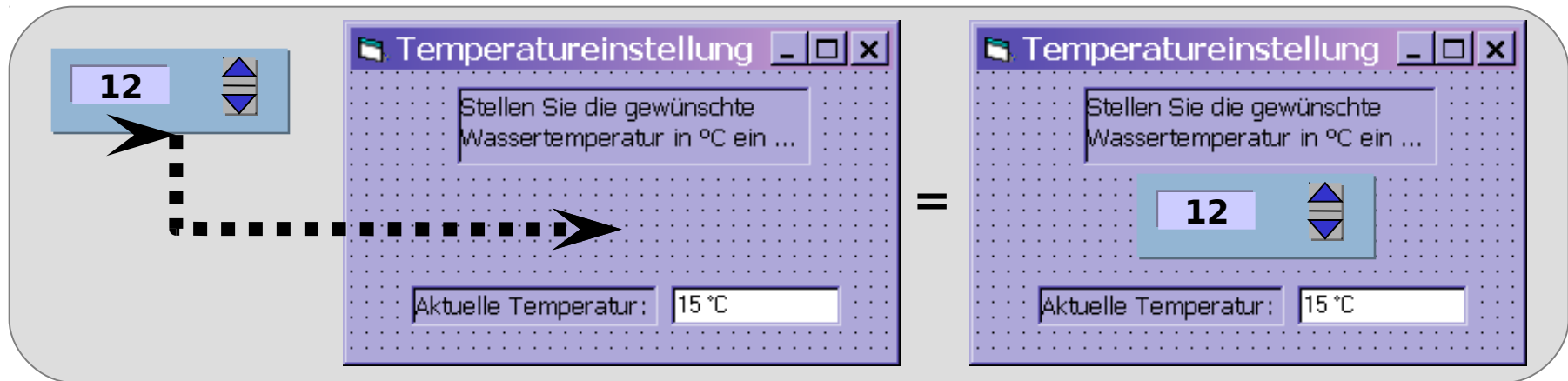


Abb: Verwenden einer Komponente

1.4. Komponentenentwicklung

Design to Component

- 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

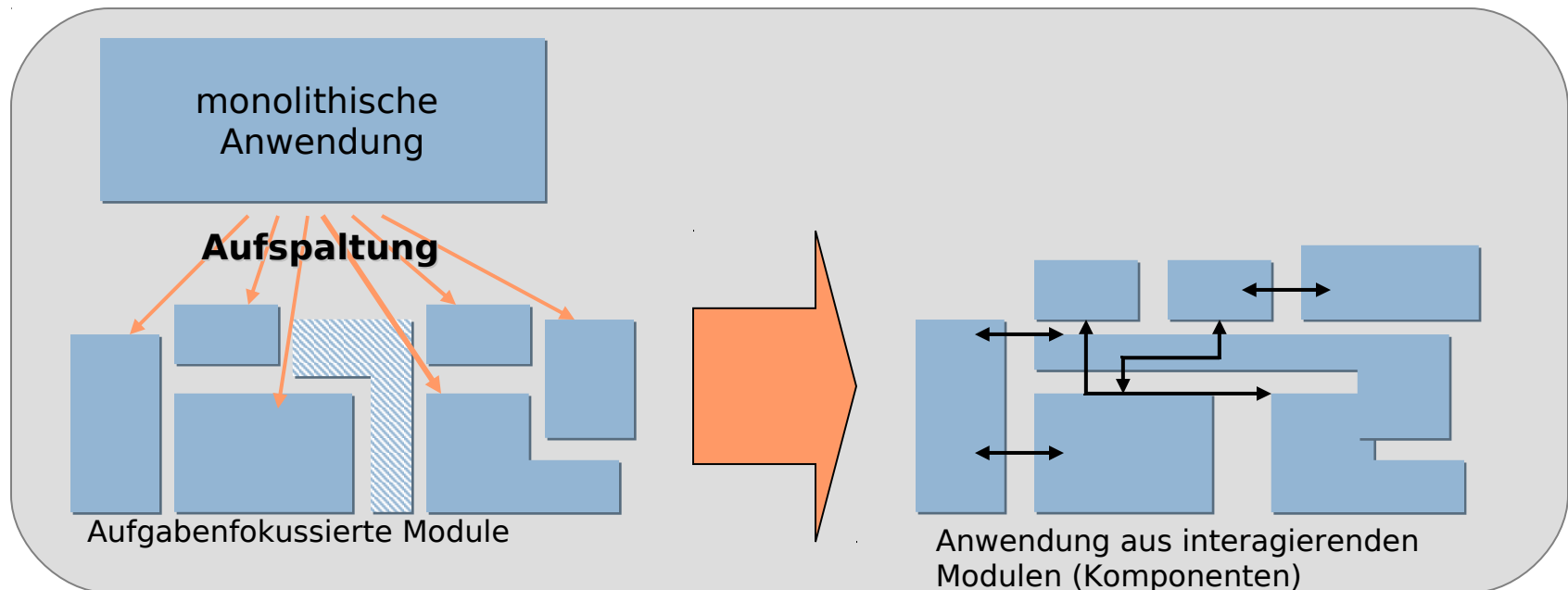
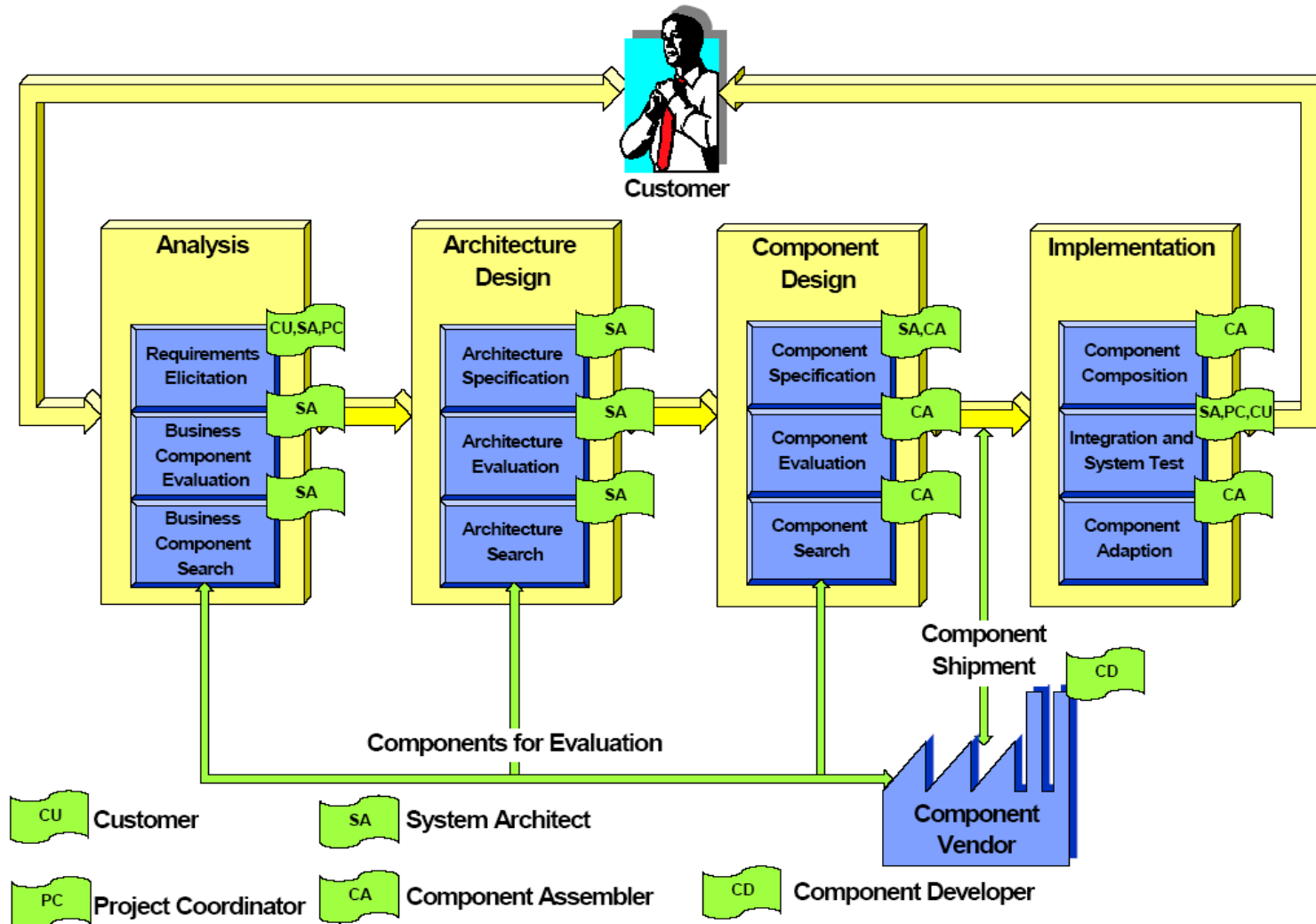


Abb: von einer monolithischen zu einer modularen Anwendung

Entwicklungszyklus für Komponenten-Software



Quelle: /Sihling 1998/

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

Was sind Komponenten und was nicht?

- „Component Software – beyond object oriented programming“
- Problem der Abgrenzung der Begriffe „Objekt“ und „Komponente“
 - synonym in manchen (Kon)texten
 - ex. auch Begriffe wie „Komponenten-Objekt“
- Objektbegriff: Instanz einer Klasse (Java) oder Clon eines Prototyp-Objekts (Factory-Pattern)
- Gemeinsamkeiten:
 - Dienste werden über Schnittstellen angesprochen
 - Schnittstellen nach Beschreibungsregeln typisiert und kategorisiert
 - Beschreibung nach Mustern (Syntax) und innerhalb von Frameworks (Semantik)
- weitere Verwirrung durch Sprachdesigner:
 - Namensräume, Moduln, Packages

- Notwendigkeit, in diesem Universum von Namen und Konzepten **aufzuräumen**, Begriffe an definierte **Bedeutungen** zu binden und in ein gewisses **Ordnungsschema** zu bringen.
- Ansatz: Bedeutungen von Begriffen durch Angabe charakteristischer Eigenschaften fixieren

Komponentendefinition

Charakteristische Eigenschaften einer Komponente:

1. Einheit unabhängiger Packung
2. Einheit der Komposition durch Dritte
3. ohne (extern beobachtbaren) Zustand

Folgerungen aus der Definition

Einheit unabhängiger Verteilbarkeit

- separiert von Umgebung und anderen Komponenten
- Kapselung charakteristischer Merkmale

Einheit unabhängiger Verteilbarkeit

- nie in Teilen zu verteilen

Komposition durch Dritte

- Verwendung ohne Kenntnis konstruktiver Details

Komposition durch Dritte

- genügend selbstabgeschlossen
- klare Spezifikation der Anforderungen und Angebote

zustandslos

- ohne diese Forderung hätten keine zwei Installationen „derselben“ Komponente garantiert gleiche Eigenschaften
- Zustand spielt sich ausschließlich auf der Ebene der Objekte ab

Zustandslos

- Kann nicht von Kopien unterschieden werden
 - Ausnahme: Zustandsparameter jenseits von Funktionalität
- Intern verwendete Zustände (oft aus Gründen der Leistungsfähigkeit) dürfen von außen nicht sichtbar sein
 - Beispiel: Cache
- Nicht sinnvoll, in einer Umgebung mehrere Instanzen einer Komponente zu halten
 - Achtung Konfusion vermeiden! Beispiel Datenbank. Unterscheide zwischen Datenbank-Server-Programm (Komponente) und darauf laufender Datenbank als „Instanz“ des Datenbank-Konzepts.
 - Diese Unterscheidung zwischen dem (fest verdrahteten) „Plan“ (= Komponente) und den (sich im Laufe der Zeit ändernden) „Instanzen“
 - Analog der Unterscheidung zwischen Objekten und Objektzuständen

Komponenten sind heute praktisch meist schwergewichtige Einheiten mit genau einer Instanz pro System