Quasar

von Tichomir Jabarski

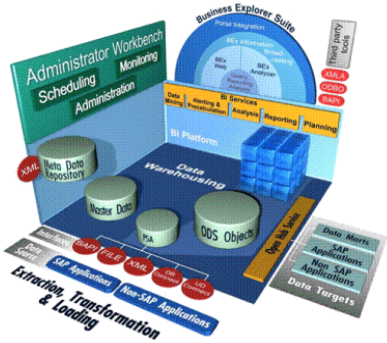
Seminar "Beiträge zum Software Engineering"

Freie Universität Berlin

Inhaltsverzeichnis

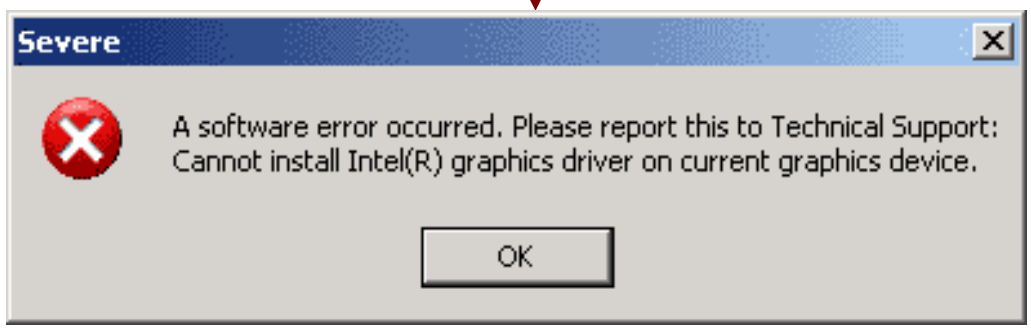
1.Quasar

Quasar I



! Schlechte Qualität !

SOFTWARE



Quasar II

Software-Projekte:

Dauern zu lange

Kosten zu viel

Scheitern oft

! Schlechte Qualität !

Was ist der Grund und hat es etwas mit Architektur zu tun?

Was ist Quasar?

QUASAR = Qualitätssoftwarearchitektur

Definiert **Verständnis für Software-Qualität**

Beschreibt wichtige **Regeln und Mechanismen der Software-technik** als Standard

Quasar auf vier Ebenen:

- Ideen & Konzepte
- Begriffe
- Standardarchitektur & Standardschnittstellen
- Standardkomponenten

Warum Quasar?

Basiert auf jahrelange Erfahrung

Definiert konkrete Praktiken in der Softwareentwicklung

Garantiert eine tragende Architektur für das Produkt

Garantiert bessere Qualität

- Architektonische Einfachkeit
- Wiederverwendbarkeit
- Entwicklungseffizienz
- Änderbarkeit

Reduziert Kosten !!!

Das Buch

Johannes, Siedersleben:

Moderne Softwarearchitektur.
Umsichtig planen, robust bauen mit Quasar

dpunkt.verlag

1 Auflage 2004

ISBN:3-89864-292-5

2. Schnittstellen

Was ist eine Schnittstelle?

Schnittstellen verbinden, schneiden nicht durch!

Die Schnittstelle formalisiert die intuitive Vorstellung wie man eine Komponente benutzt

- System verständlicher
- Abhängigkeiten Reduzieren
- Arbeit sparen durch Wiederverwendung

Schnittstellen sind die Träger der Architektur

Viele Muster sind ein Spezialfall des Schnittstellen-Gedankens

- Brücke, Strategie, Fabrik, Iterator

Typen von Schnittstellen

Standard-SS

Allgemein akzeptiert

Gut dokumentiert

Gut getestet

java.lang.*, JDBC, u.s.w

Angebotene-SS

Implementierung existiert

Importeur für die SS gebaut

Enge Kopplung

← *Adapter* →

Angeforderte-SS

SS wird an den Importeur angepasst

Lose Kopplung

Verfeinerung von Schnittstellen

Syntax (Rückgabewerte, Argumente, in/out, Typen)

Semantik (Was bedeutet die Methode)

- Formell – OCL, JML, VDM, Z, QSL
- Informell – z.B JavaDoc

Protokoll (z.B synchron, asynchron)

Nicht funktionale Eigenschaften (Performance, Robustheit, Verfügbarkeit)

Semantik von Schnittstellen

2 Arten von Methoden:

- Kommandos – ändern den Zustand des Systems
- Abfragen – lassen das System unverändert
  - Einfache Abfragen
  - Abgeleitete Abfragen

Semantik:

- Was bewirken die Kommandos
- Was bedeuten die Abfragen

! Die meisten Schnittstellen sind komplizierter als sie aussehen !

Schnittstellen und Entwicklungsprozess

Wer muss was wissen?

SS, die alle Entwickler sehen, müssen sehr einfach sein

komplexe SS sollen nur für wenige Entwickler sichtbar sein

Die Schnittstellen-Architektur beeinflusst den Entwicklungsprozess sehr:

Parallelisierung der Entwicklung

Verteilung des Knowhow innerhalb des Teams

Rollenzuweisung im Team

3. Komponenten

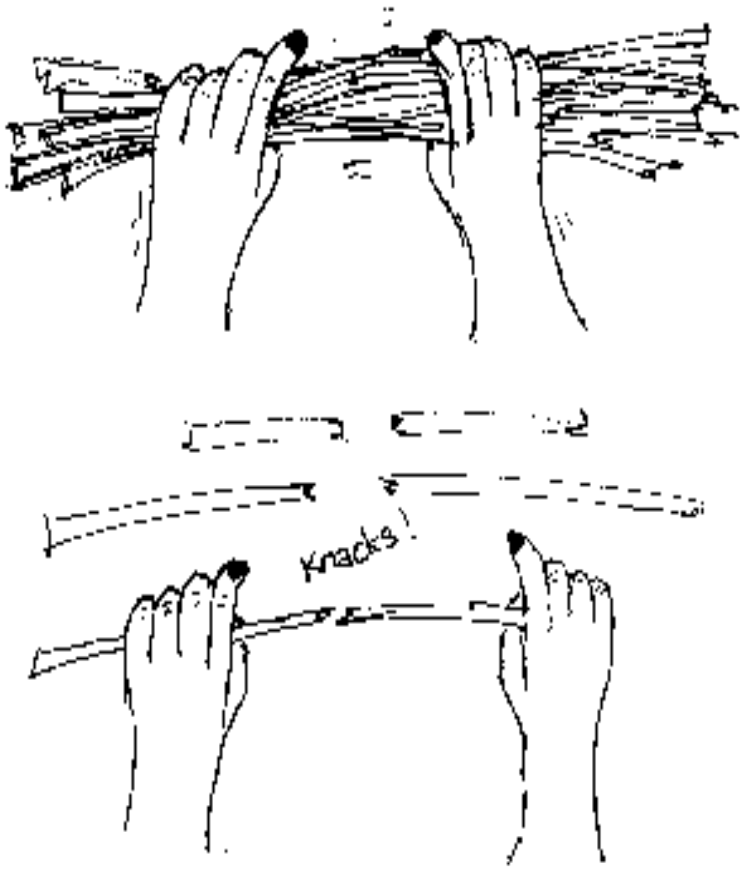
Was ist eine Komponente?

Der einzige Weg Systeme zu entwerfen und zu verstehen

„Teile und herrsche“ Prinzip

Merkmale:

- Exportiert mind. 1 SS
- Importiert andere SS
- Versteckt Implementierung
- Minimale Annahmen über die Umgebung
- Komponenten lassen sich komponieren



Komponenten & Objektorientierung I

? Klasse = Komponente

Klasse als Entwurfseinheit zu klein

Man kann ohne OO hervorragende Komponenten bauen

Mit OO können auch sehr schlechte monolithische Architekturen entstehen

Naive Objektorientierung führt oft zu unwartbaren Systemen, in denen tausende Klassen wahllos miteinander kommunizieren

Datenkapselung erreicht man durch Verstecken von Klassen und nicht durch Verstecken von Feldern

Komponenten & Objektorientierung II

Strikte Kontrolle der Importanweisungen: wichtigste Maßnahme beim Entwurf

Zyklen strikt vermeiden

OO nur innerhalb Komponenten

Komponenten kommunizieren ausschließlich über SS

- Vererbung über Komponentengrenzen strikt verboten
- Anwendungsbeziehungen nur beschränkt mit Objektreferenzen implementierbar

Konfiguration

Wer verbindet Benutzer und Implementierung?

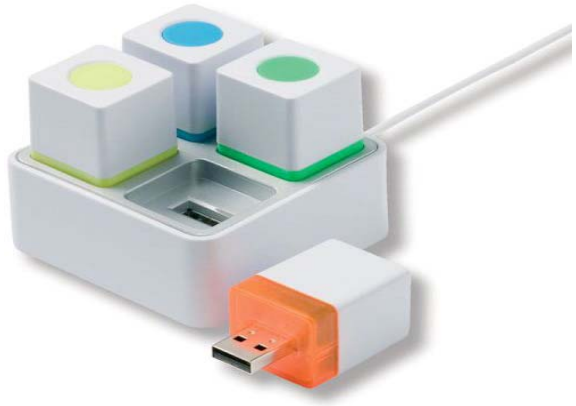
- Importeur und Exporteur kennen sich nicht
- Die Konfiguration verbindet die beiden

Legt die konkrete Implementierung fest

- Zur Compilezeit
- Zur Laufzeit

Möglichkeiten für Konfiguration

- Konstruktor
- Fabrik
- Namensdienst



! Konfiguration nur an bestimmten Stellen !

Kompositionsmanager

Stellt eine geeignete Umgebung für die Komponente

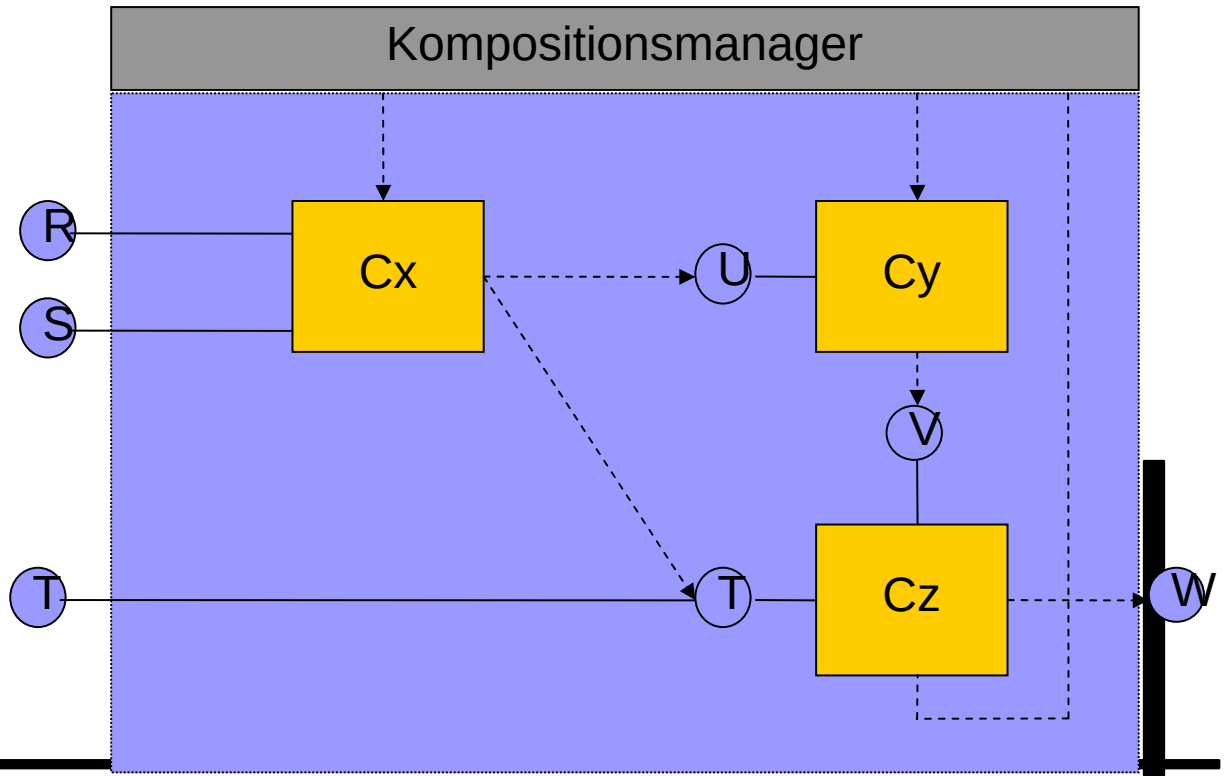
- Konfiguration
- Ausnahmebehandlung
- Bereitstellung von Ressourcen (Threads, Sockets, u.s.w), Aufbau von Verbindungen, Transaktionskontrolle

Liefert eine Komposition

Exportiert eine Teilmenge aller importierten SS

Fassade für die exportierten SS

Konfiguration schränkt Freiheitsgrade ein



Wie findet man Komponenten?

Entwurf der Schnittstellen

Validieren der Schnittstellen

Implementieren der Komponente

Validieren der Implementation

Komponente verbinden durch Konfiguration

Wie beschreibt man Komponenten?

Beschreibung von außen nach innen:

- Übersicht: Idee hinter der Komponente, ein bis zwei Seiten
- Außensicht: alle Informationen für den Benutzer
- Innensicht: inneren Aufbau der Komponente für die Entwickler
- Variabilitätsanalyse: Anwendungsfälle für evtl. Änderbarkeit

Häufige Fehler :

- gar keine Beschreibung
- unsystematisch
- zu detailliert

4. Softwarekategorien

Was ist Softwarekategorie?

Softwarekategorien ⟷ **Sachgebiete menschlichen Wissens**

Software „kann & weiß“ etwas

- Softwareblutgruppe
- Softwarekategorie

Komponenten finden ist schwierig

Kategorien finden ist einfacher

Analysieren der Kategorien

Zerlegung in Komponenten

Zuordnung von Schnittstellen und Komponenten zu Kategorien

Typen von Softwarekategorien

Hauptkategorien:

0-Kategorie: allgemein bekannte Komponenten & Schnittstellen

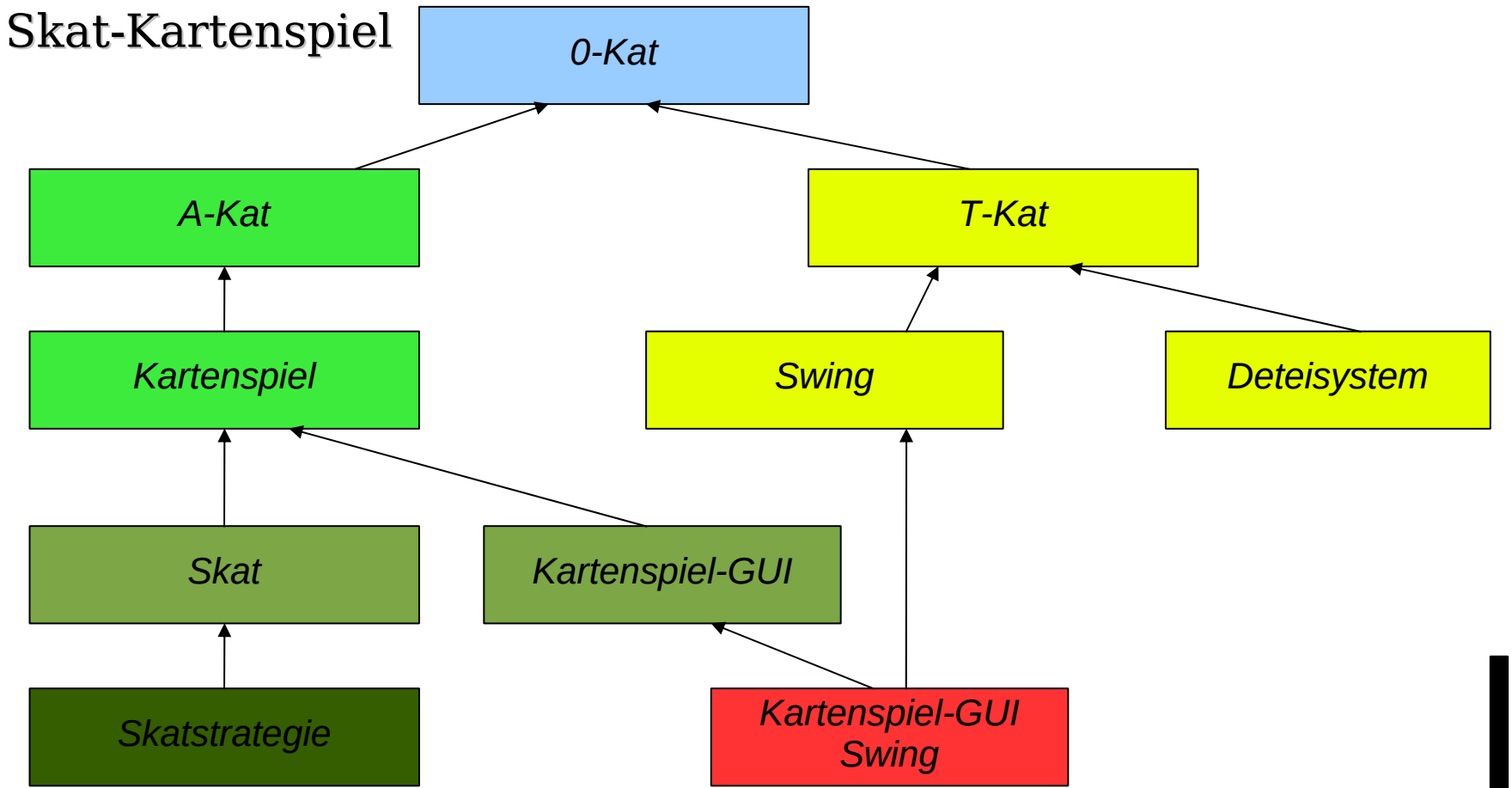
A-Kategorie: Anwendungssoftware frei von technischen Aspekten

T-Kategorie: technische Komponenten & Schnittstellen

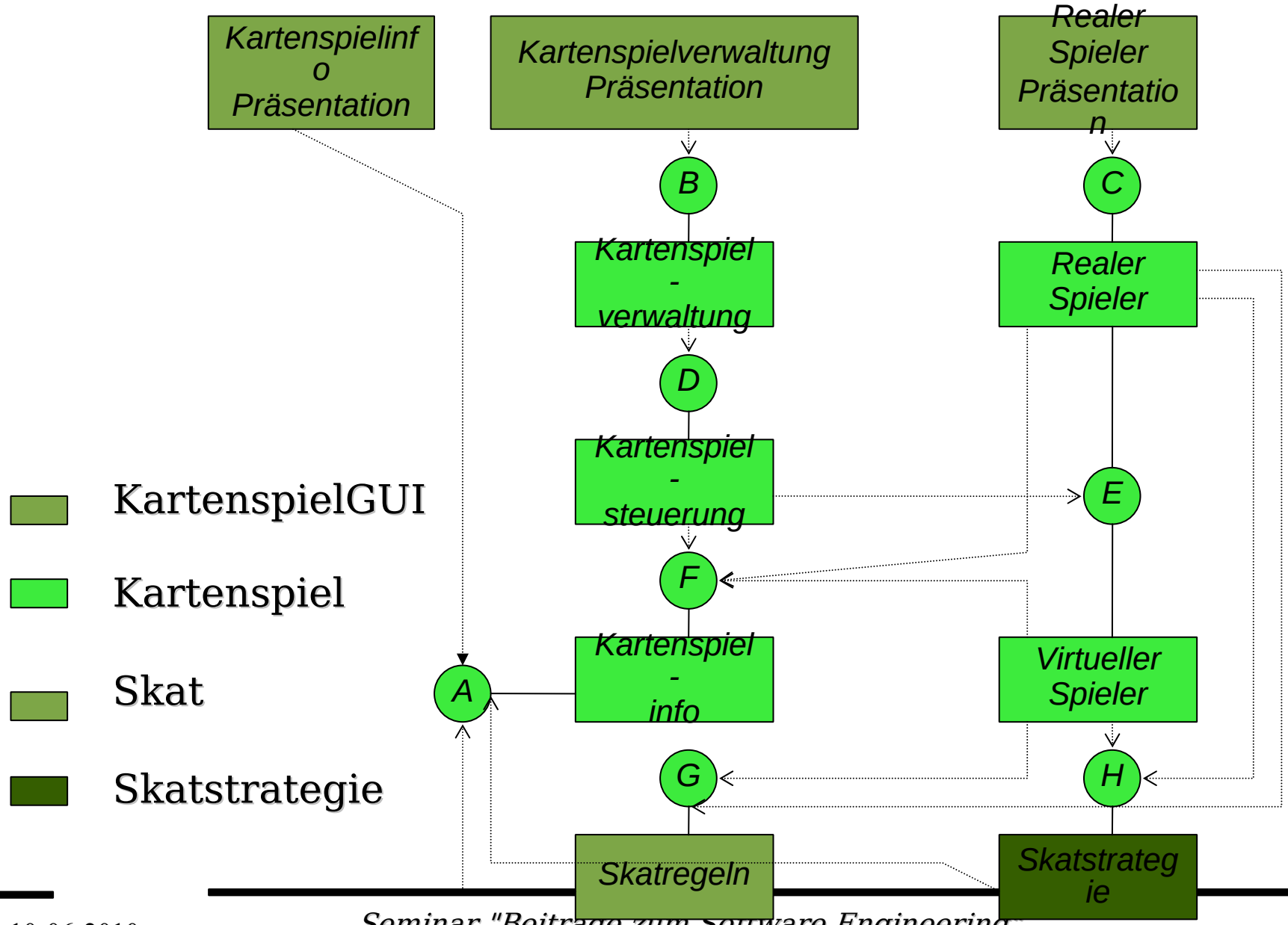
R-Kategorie: Adapter zwischen Kategorien

AT: Vermischung von Anwendung und Technik

Beispiel für Softwarekategorien



Beispiel für Softwarekategorien



Kommunikation zwischen Komponenten

0-Kat darf von allen verwendet werden

Komponente verschiedener Kategorien kommunizieren nur über den ersten gemeinsamen Vorfahren im Konfigurationsgraphen

Komponenten können kommunizieren sobald sie eine gemeinsame Gesprächsebene besitzen und zwar in Form einer Schnittstelle der passenden Kategorie

Eine Komponente der Kategorie X darf eine Schnittstelle XY der Kategorie Y genau dann importieren oder exportieren wenn X eine Verfeinerung von Y ist

!Vermischung von Anwendung und Technik führt zu unwartbaren Systemen !

Komplexitätsmaß

Reine Kategorie: im Kategoriegraphen gibt es nur einen Weg zur 0-Kat

Unreine Kategorie: vermengt zwei oder mehrere Kategorien

Komplexitätsmaß:

Rein: Anzahl der Vorfahren im Graphen

Unrein: Summe der Komplexität aller oberen Kategorien

Modul: Komplexität der Kategorie zu der es gehört

Komponente: Gewichtete Mittel der Komplexitäten aller Subkomponenten

A- und T- Revolution oder Selbstverständlichkeit?

Trennung von Anwendung und Technik ist allgemein akzeptiert

Fachlogik und Technologien haben verschiedene Lebenszyklen

> „Software, die sich unterschiedlich schnell ändert, sollte in verschiedene Module untergebracht werden [Parnas 72].“

Kritik am Konzept der Softwarekategorien:

- kostet Performance ↔ Gewinn an Flexibilität
- Aufwand beim Entwurf ↔ einfacher bei der Entwicklung
- Überflüssig ↔ bessere Änderbarkeit und Stabilität

5. Zusammenfassung

Was sollte man vom Vortrag mitnehmen?

- Softwarearchitektur ist der wichtigste Erfolgsfaktor einer Software
- Schnittstellen als Träger der Architektur
  - Typen von Schnittstellen
  - Information in der Schnittstelle
- Komponenten
  - Komponentenorientierung und OO nicht verwechseln
  - Konfiguration und Kompositionsmanager verwenden
  - Beschreibung der Komponenten
- Softwarekategorien
  - Schubladen zum Kategorisieren des Knowhos der Software
  - Typen von Softwarekategorien
  - Halten die Abhängigkeiten konstant

Weitere Themen

Fehler & Ausnahmen

Spezifikation von Schnittstellen

Softwarearchitekturen

Anwendungskomponenten & Anwendungskern

Quasar Ende

Vielen Dank fürs Zuhören!

… und lesen Sie das Buch! Es ist wirklich empfehlenswert!