Seminar „Ausgewählte Beiträge zum Software Engineering“

Softwaredesignmethoden

Irina Gimpeliovskaja

- Softwaredesignmethoden
- Eigenschaften
- Was sie bewirken
- D-Matrix
- Ausarbeitung und Transformation der Matrix

Quellen

- David Budgen: Software Design, Pearson Education Limited, 2003 (Second edition)
  - Kapitel 8, 9
- http://alistapart.com/articles/method/

Softwaredesignmethode

- Was ist eine Softwaredesignmethode?
  - Prozedurale Beschreibung des Designens
  - Beschreibt die Aufgaben und deren Reihenfolge
  - Nicht Problem-spezifisch
  - Produktion einer Methode für die Lösung vom Designer
  - Einzigartige Rolle in der Softwareentwicklung

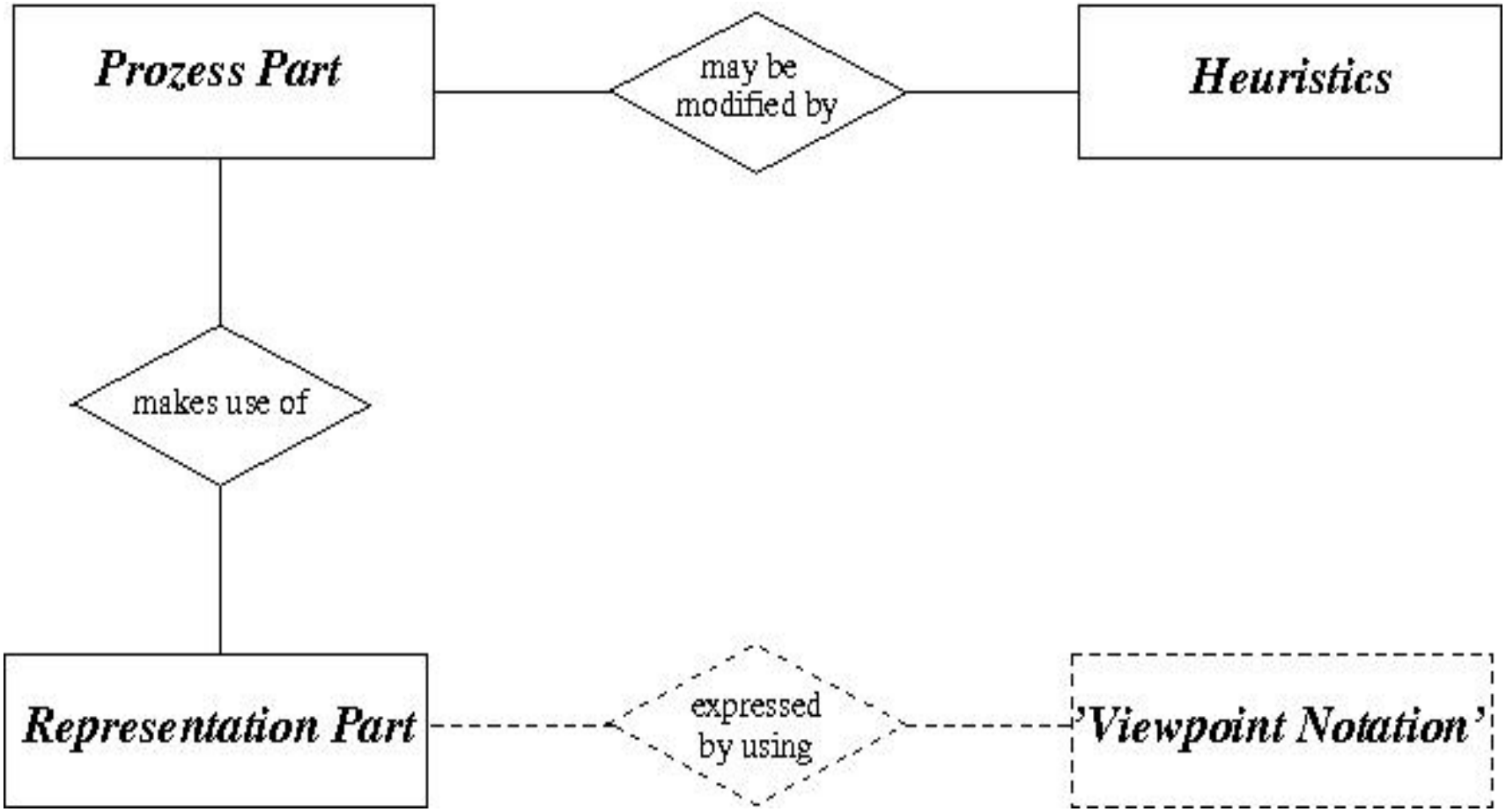
Eigenschaften von Softwaredesignmethoden

- Eigenschaften der SDM
  - Nicht problem-spezifisch
  - Wenig verordnend
  - Identifiziert allgemeine Strategie
- Was bewirken die Softwaredesignmethoden?
  - Vermeidung ständiger Prüfung
  - Bestimmung von Deadlines
  - Konsistenz
  - Peer-to-peer Übertragung für Praktiker
  - Dokumentation des Designprozesses

Wissen und Bestandteile der SDM

- Vessey und Conger teilten das Wissen ein
  - Deklaratives Wissen
    - Benennung der Aufgaben
  - Prozedurales Wissen
    - Ausführung in aktueller Situation
- Bestandteile der SDM
  - Repräsentationsteil
    - Notation mit einer oder mehreren Perspektiven
  - Prozessteil
    - Prozeduren und Strategien zum Vorgehen
  - Heuristiken
    - Anpassung für aktuelles Problem
    - Basiert auf Erfahrung

Softwaredesignmethoden bestehen aus folgenden Komponenten:



Kategorien

- Softwaredesign Methoden werden auf folgende Kategorien aufgeteilt:
  - Formale Methoden
    - Benutzen mathematische Schreibweise
    - Vorteil: Konsistenz
    - Prozessteil wenig entwickelt
  - Systematische Methoden
    - Weniger mathematisch
    - Repräsentationsteil meist in Form von Diagrammen
    - Größerer Anwendungsbereich

Was die SDM im Softwareentwicklungsprozess bewirken?

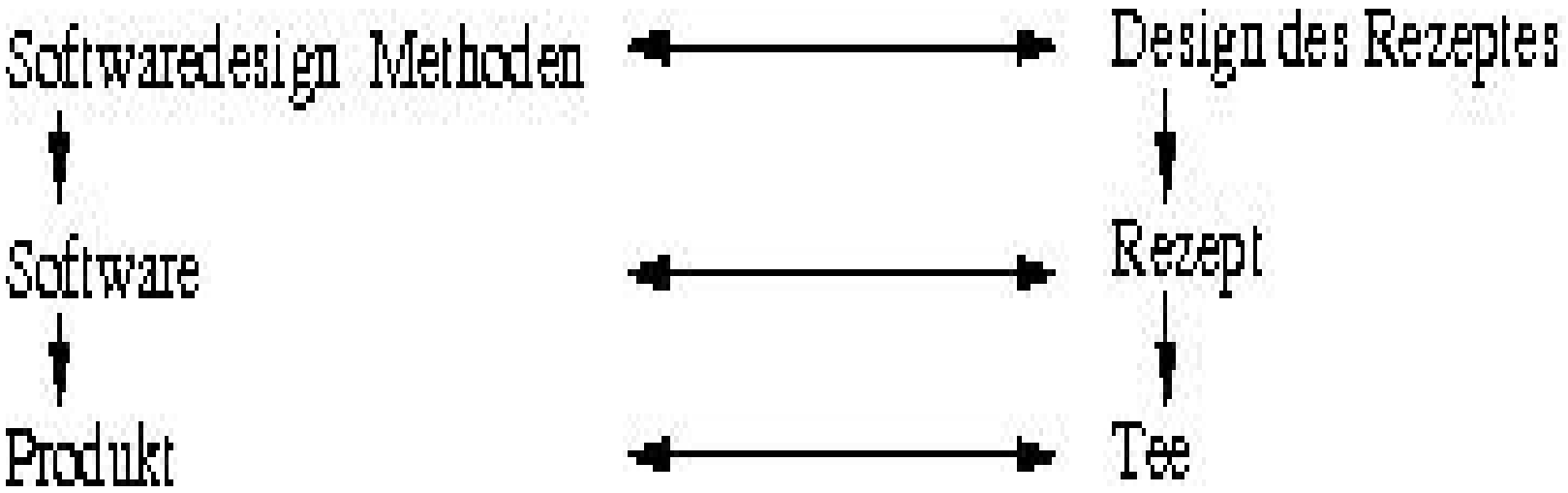
- Zwei Aspekte des Softwareentwicklungsprozesses profitieren davon
  - Technische Aufgaben
    - Wissensvermittlung
    - Konsistent strukturiertes System
    - Standardisierte Form für Repräsentationen
    - Vermeidung häufiger Fehler
  - Aufgaben des Managements
      - verbunden mit technischen Aufgaben
    - Framework zum Abspeichern von Entscheidungen
    - Bessere Abschätzung
    - Framework für Meilensteine
- SDM helfen bei der Strukturierung des Designprozesses und Designproduktes

Wieso bewirken die SDM keine Wunder?

- Eine Softwaredesignmethode liefert nur ein Framework
  - Ratschläge für Repräsentationen
  - Anleitungen auf Schritten
  - Kriterien zum Nachdenken
- Weiteres entscheidet das konkrete Problems
  - Bsp: Design der Software wie Design des Rezeptes
- Was die Softwaredesignmethode nicht bewirkt:
  - Es kann nicht spezifischer werden
  - Designer muss sich in der Domäne des Problems auskennen
  - Liefert Begriffe, Beschreibungen, Präsentation, aber keine problem-spezifischen Lösungen
  - Designmethode ist selbst kein Rezept, das man blind befolgen kann

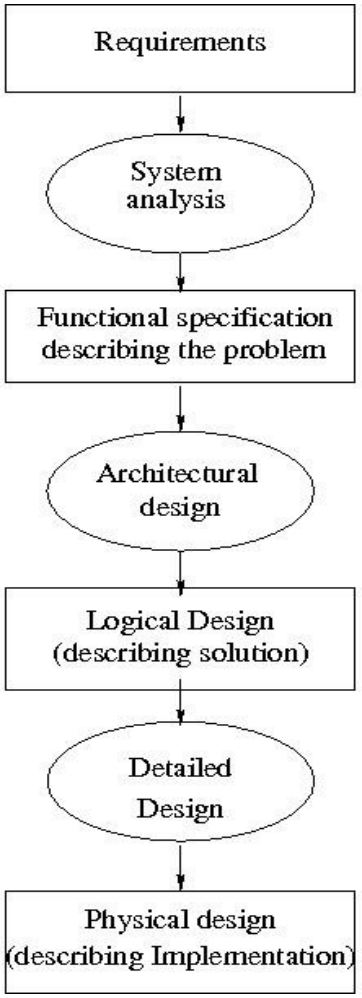
Beispiel mit Tee

- Designmethode ist kein Rezept!

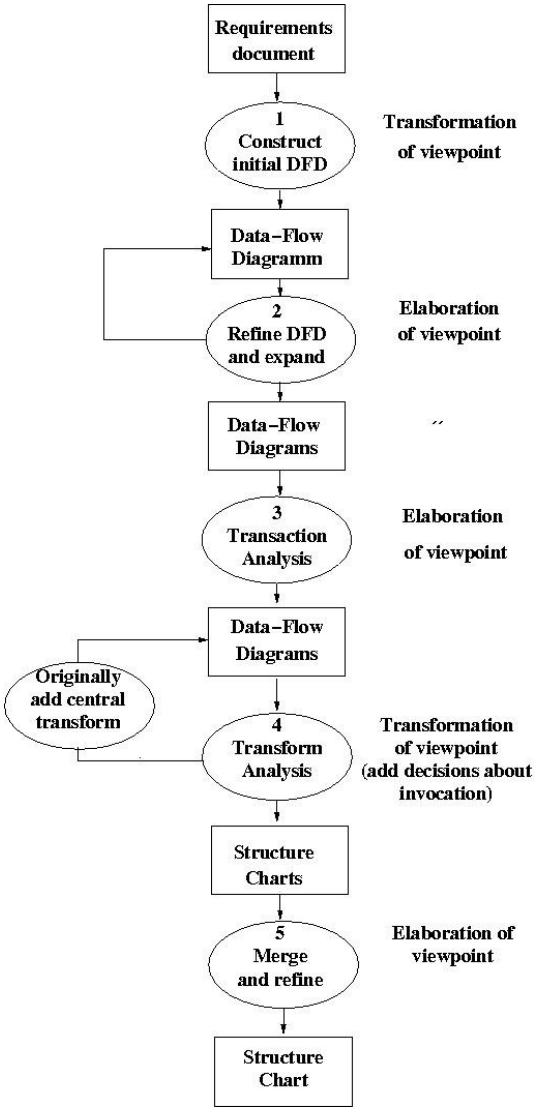


Beispiel einer Methode

- Wie oben erwähnt bestehen die SDM aus:
  - Repräsentationsteil
  - Prozessteil
  - Heuristiken
- Das Bild zeigt Prozessteil einer Methode



Größeres Beispiel



- Transformationsdiagramm für SSA/SD (Structured System Analysis and Structured Design)

Form des typischen Prozessteils

- Zwei Arten von Schritten:
  - Transformationsschritt
    - Modifikation der Struktur der Modells
    - Oft Wechsel der Perspektive
  - Ausarbeitungsschritt
    - Umorganisation des Modells innerhalb der Perspektive
    - Hinzufügung weiterer Informationen, bessere Einsicht
- Gruppierung der Softwaredesign Methoden:
  - Dekompositionsmethoden (top-down)
  - Kompositionsmethoden (bottom-up)
  - Organisationsorientierte Methoden
  - Template-basierte Methoden

D(esign)-Matrix

- Abstrakte Beschreibung des Softwaredesignprozessteils
  - Vereinigung der Transformations- und Ausarbeitungsschritte
- Annahme: jedes Design kann durch Spezifikation von Designelementen beschrieben werden
  - Unteraufgaben, Anwendungen, Klassen
- Elemente haben 4 Perspektiven (viewpoints):
  - Funktion (f)
  - Verhalten (b)
  - Datenmodell (d)
  - Konstruktion (c)
- Spalten für Elemente, Zeilen für Viewpoints

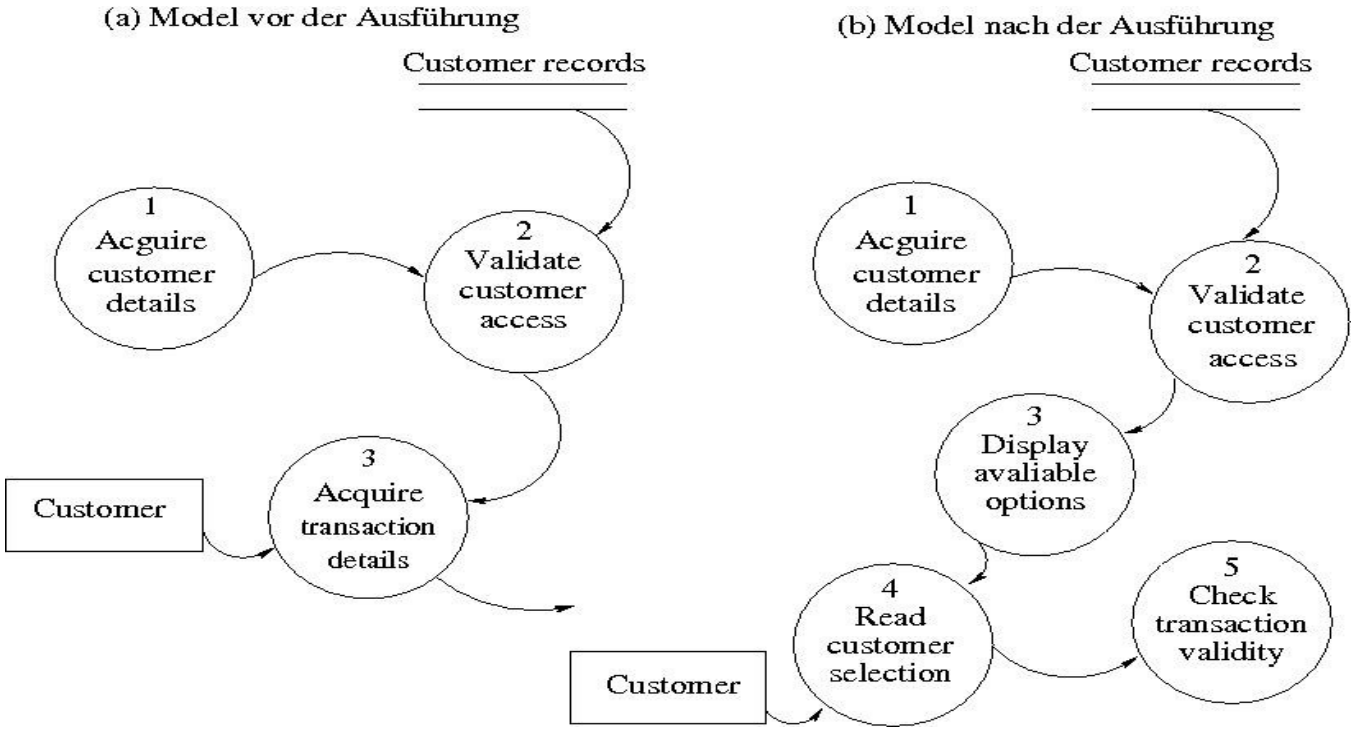
$$\begin{bmatrix} D_1^b & D_2^b & \cdots & D_n^b \\ D_1^f & D_2^f & \cdots & D_n^f \\ D_1^d & D_2^d & \cdots & D_n^d \\ D_1^c & D_2^c & \cdots & D_n^c \end{bmatrix}$$

D-Matrix 2

- Am Anfang nur Anforderungsspezifikation
- Im Verlauf des Designs kommen Elemente hinzu
  - $D_i^b \quad D_{i1}^b \quad D_{i2}^b \quad D_{i3}^b$
  - $D_i^f \leftarrow D_{i1}^f \quad D_{i2}^f \quad D_{i3}^f$
  - $D_i^d \quad D_{i1}^d \quad D_{i2}^d \quad D_{i3}^d$
  - $D_i^c \quad D_{i1}^c \quad D_{i2}^c \quad D_{i3}^c$
- Der Designprozess ist dann die Transformation der Startmatrix in die Zielmatrix
- $$\begin{pmatrix} R^b \\ R^f \\ \not{O}^d \\ \not{O}^c \end{pmatrix} \rightarrow \begin{pmatrix} D_{i1}^b & D_{i2}^b & D_{i3}^b \\ D_{i1}^f & D_{i2}^f & D_{i3}^f \\ D_{i1}^d & D_{i2}^d & D_{i3}^d \\ D_{i1}^c & D_{i2}^c & D_{i3}^c \end{pmatrix}$$

$$\begin{pmatrix} R^b \\ R^f \\ \not{O}^d \\ \not{O}^c \end{pmatrix}$$

Beispiel Ausarbeitungsschritt

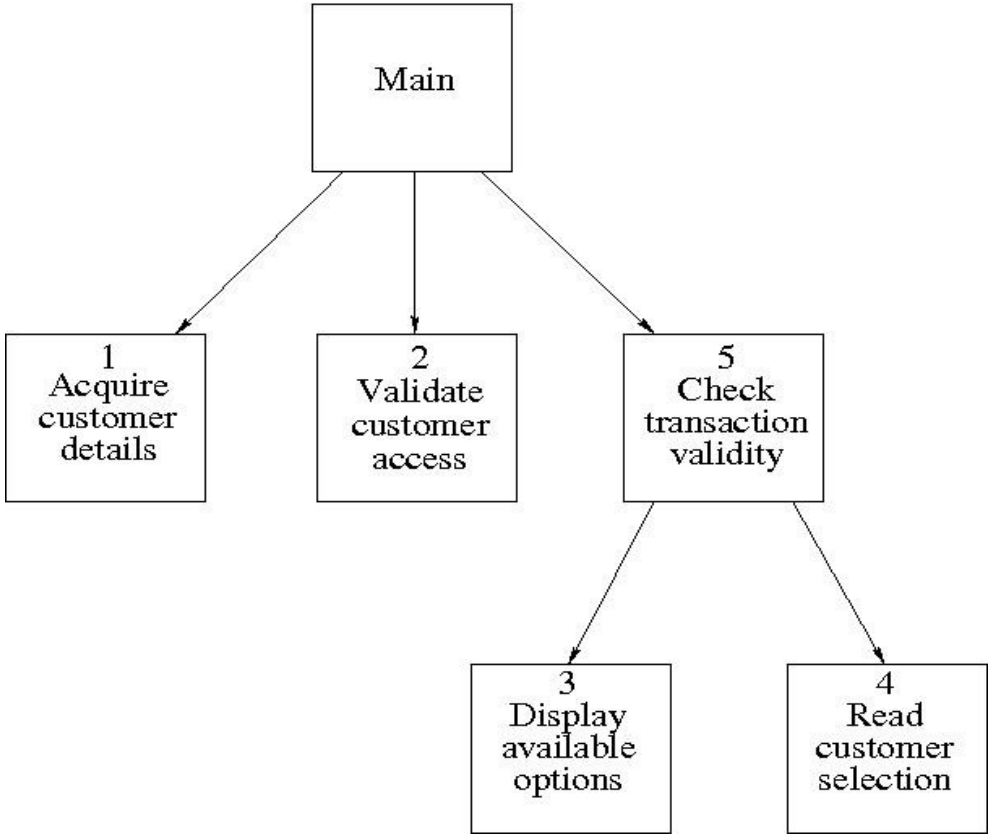


$$\begin{bmatrix} \varnothing_1^b & \varnothing_2^b & \varnothing_3^b \\ D_1^f & D_2^f & D_3^f \\ \varnothing_1^d & \varnothing_2^d & \varnothing_3^d \\ \varnothing_1^c & \varnothing_2^c & \varnothing_3^c \end{bmatrix} E^3 \rightarrow \begin{bmatrix} \varnothing_1^b & \varnothing_2^b & 0_3^b & \varnothing_4^b & \varnothing_5^b \\ D_1^f & D_2^f & D_3^f & D_4^f & D_5^f \\ \varnothing_1^d & \varnothing_2^d & \varnothing_3^d & \varnothing_4^d & \varnothing_5^d \\ \varnothing_1^c & \varnothing_2^c & \varnothing_3^c & \varnothing_4^c & \varnothing_5^c \end{bmatrix}$$

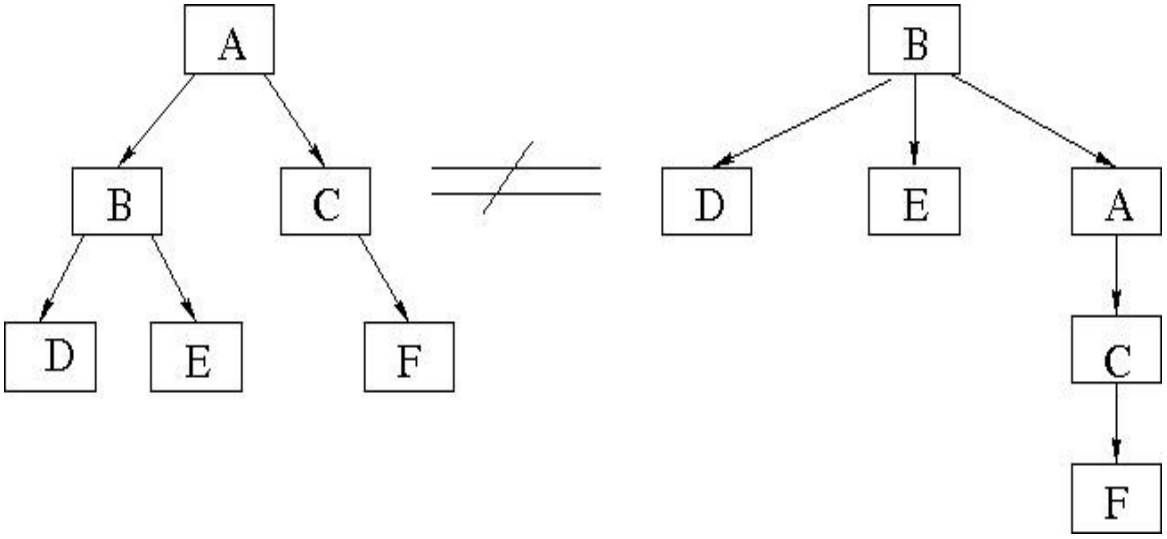
Beispiel Transformationsschritt

$$\begin{bmatrix} \emptyset_1^b & \emptyset_2^b & \emptyset_3^b & \emptyset_4^b & \emptyset_5^b \\ D_1^f & D_2^f & D_3^f & D_4^f & D_5^f \\ \emptyset_1^d & \emptyset_2^d & \emptyset_3^d & \emptyset_4^d & \emptyset_5^d \\ \emptyset_1^c & \emptyset_2^c & \emptyset_3^c & \emptyset_4^c & \emptyset_5^c \end{bmatrix}$$

$$T^4 \rightarrow \begin{bmatrix} \emptyset_1^b & \emptyset_2^b & \emptyset_3^b & \emptyset_4^b & \emptyset_5^b \\ D_1^f & D_2^f & D_3^f & D_4^f & D_5^f \\ \emptyset_1^d & \emptyset_2^d & \emptyset_3^d & \emptyset_4^d & \emptyset_5^d \\ D_1^c & D_2^c & D_3^c & D_4^c & D_5^c \end{bmatrix}$$



Design bei Dekomposition



$$\begin{bmatrix} \varnothing_1^b & \varnothing_2^b & \varnothing_3^b \\ D_1^f & D_2^f & D_3^f \\ \varnothing_1^d & \varnothing_2^d & \varnothing_3^d \\ D_1^c & D_2^c & D_3^c \end{bmatrix} E^n \rightarrow \begin{bmatrix} \varnothing_1^b & \varnothing_2^b & \cdots & \varnothing_n^b \\ D_1^f & D_2^f & \cdots & D_n^f \\ \varnothing_1^d & \varnothing_2^d & \cdots & \varnothing_n^d \\ D_1^c & D_2^c & \cdots & D_n^c \end{bmatrix}$$

Zum Buch vom Budgen

- Inhalt:
  - Vor allem D-Matrizen bewirken abstrakte Herangehensweise
  - Man versteht besser wozu so eine Anzahl von Perspektiven, Modellen, etc existiert
  - Man denkt über Softwaredesign anders
- Darstellungsweise:
  - Schöne Beispiele
  - Die Erklärung ein bisschen verwirrend
  - Zum lernen sollte man ein zweites Buch dazu holen ;)

Danke!