

AG Designprozess

Where is the beef?

Drink the cool aid.

2 – Methoden

- Bottom-up
 - am Beispiel den Prozess selber erfinden
- Top-down
 - abstrakten Design-Prozess am Beispiel erläutern

Akzeptanz (I)

- Gutes Design macht Programme nachvollziehbar und (evtl.) beweisbar.
- Design Rezepte:
 - Vorteil, weil Studierende wissen, wie sie an ein Problem herangehen
 - Nogo für Studierende, die kreativ eigene Programme entwickeln wollen
- „Erst durch die Hölle gehen“, um den Sinn zu verstehen.
 - an ein Problem heranzuführen, das keiner ohne Designprozess lösen kann.
 - „schwieriges Problem“: mutual recursion (auch schlimm für Hacker). Bsp.: Interpreter / Object-Serializer
 - DoD Competition (Abwandlung Shape Beispiel)
 - Haskell: 98 loc

Das Lehren eines Designprozesses benötigt eine einfache Sprache

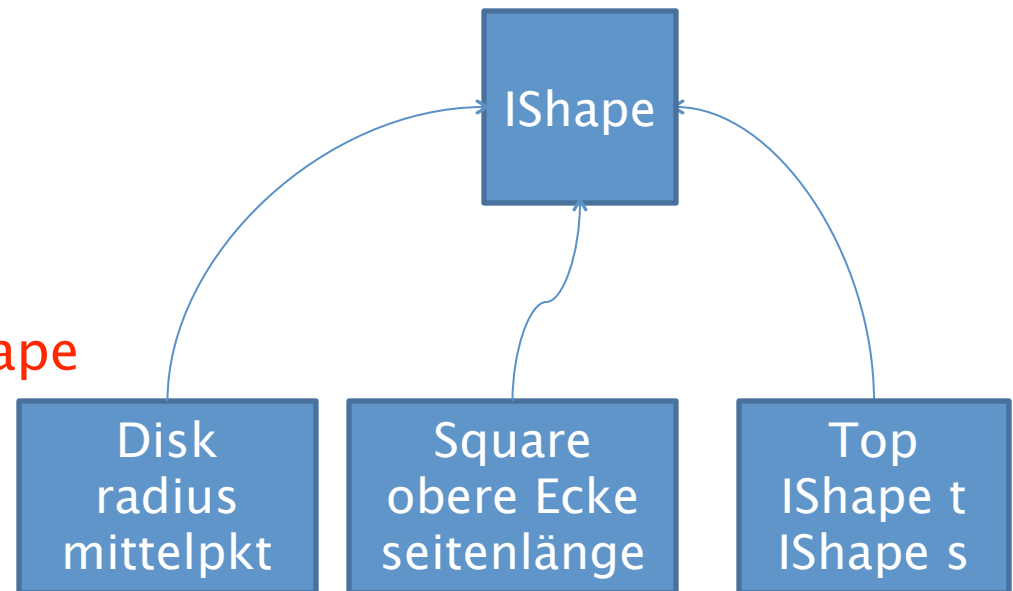
- jede existierende marktrelevante Sprache wird die Studenten überfordern.
- man braucht eine kleine und einfache Sprache, um die Konzepte des Designs zu verstehen und anzuwenden.
- man muss sich entscheiden
 - Fokus auf dem Designprozess
 - Fokus auf dem Erlernen einer „marktgängigen“ Programmiersprache

Designprozess (I) – von INFO zu DATA

- INFO

- Shape is one of
 - disk
 - square
 - a shape on top of a shape

- DATA



Was ist wichtiger Datenstrukturen oder Algorithmen?

- 2 Rekursion
 - strukturell
 - generativ
- strukturell
 - cons insertion sort (**traversal** und **insert**)
- generativ (eine weitere Instanz vom gleichen Problem) merge, quick
 - template (triviales Problem und triviale Lösung)
 - generate – Rekursion
 - combine – Zusammenfügen
- muss man die zweite Art der Rekursion in den Anfängervorlesungen vorstellen?

Problem Spezifikation / Implementation und schneller Wechsel zwischen den Ebenen

- Spezifikation:
 - was muss das Programm erledigen?
- Implementation:
 - wie muss das Programm arbeiten?

Schwierig ist der häufige Wechsel zwischen den Ebenen, der eingeübt werden muss.

Fin

Wie transportiere ich, dass eine Spezifikation widerspruchsfrei und / oder vollständig sind

- Gleichungen spezifizieren

- final interpretation
- initial interpretation

- 80'er

constructive algebraic
specification (first order logic)
design recipe bewegt sich auf
first order logic

KEIN IO in der

Beispiele

- Under specification : Addiere die folgenden Zahlen zusammen:
- Over specification: Generative Rekursion : Finde den kürzesten Pfad in einem Graphen, wenn es mehr als zwei Pfade gibt.
- Inconsistent specification: Stabiler Quicksort-Algorithmus

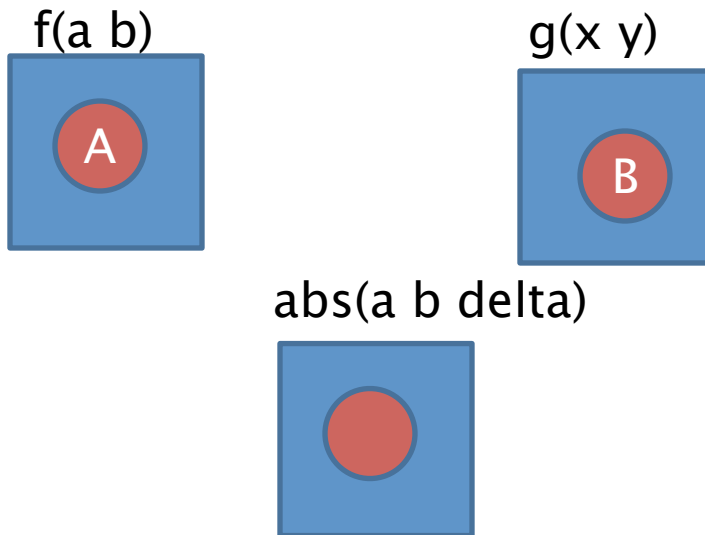
Sortieren vollständig spezifizieren:
keine Elemente weglassen
keine Elemente hinzufügen
Keine Elemente vertauschen

Designprozess

- Zu Beginn kein expliziter Zustand in Programmen.
- Domain-Wertklassen für physikalische Dimensionen (Zeit, Währung) reduzieren Code.
- atomic data (Skalar, Vektor, Betrag, Punkt)
- compound data (Matrix, ...)
 - aus Sicht der Mathematik ist compound data atomic data

Warum funktioniert objektorientiertes Programmieren nicht?

- Abstraktion???????



Warum funktioniert objektorientiertes Programmieren nicht?

- **Shape:**
 - traversal mit **in?**
 - traversal mit **render?**
 - traversal mit **area?**
- **Visitor**-Pattern ist funktional
Referenz auf **this/self** wird auf den **Visitor** übergeben

- Man verlässt sich darauf, dass das Programm korrekt ist.
Performance Test.
- Mit **Visitor** oder **Composite** in z.B. Java, C++, Ruby.
- Tail Call Optimization sind in Scheme / Lisp aber nicht in OO Sprachen implementiert
- In allen gängigen OO Sprachen muss TCO mühsam interpretiert werden.

Designprozess (II) – Behaviour on Data – am Beispiel

- **check-expect**
 - Verhalten in das Interface
- **in?** für alle Klassen implementieren
- **Top** (Nutzen der Selbstreferenz)
- **Top**, daraus fällt dann das Composite Pattern
- Doktoranden
Shapeproblem lösen

Algorithmen versus Daten von hinten programmieren

- Von hinten programmieren
- Wenn das Problem einfacher wäre, dann kann ich es lösen.
- Was sind die preconditions, die dann zu den postconditions des vorherigen Schrittes werden.
- Beispiel für notwendiges „rückwärts Programmieren“.
- `parse ( XML -> s-expr )`.
- Ziel abstrakter Syntaxbaum für einen Shape.
- Datentyp des Outputs bestimmt die Verarbeitung.

ISO– language independent datatypes

- primitive: (Date, Time, Number)
- existential Type (Typ existiert)