Dokumentation von Entwurfsmustern

Zwei kontrollierte Experimente über die Nützlichkeit der Dokumentation von Entwurfsmustern bei der Programmwartung

Inhalt

- Dokumentation von Entwurfsmustern
- Experiment
  - Motivation, Beschreibung, Ergebnisse und Auswertung
- Bewertung
  - Experiment, Aufsatz, Zusammenfassung

Dokumentation von Entwurfsmustern

Entwurfsmuster

- Programmqualität erhöhen
- Programmierproduktivität erhöhen
- Verbesserung der Kommunikation zwischen Entwicklung und Wartung
- Beispiele: Kompositum, Besucher, Beobachter, Schablonenmethode

Dokumentation

- Pattern Comment Line (PCL)
- zusätzliche Kommentare
- Angabe verwendeter Entwurfsmuster
- “Welche Klasse hat welche Funktion gemäß Entwurfsmuster?”

Experiment

Motivation

- Was will man untersuchen?
  - Nützlichkeit der Dokumentation von Entwurfsmustern bei Wartung von Programmen

Motivation

- Wieso will man das untersuchen?
  - Nützlichkeit von Entwurfsmustern wird immer unterstellt, aber ist nicht wissenschaftlich bewiesen

Exkurs - Wie versteht man Programme?

- Aus Teilen aufs Ganze schließen (Hierarchie von Hypothesen)
- Entwurfsmuster: Ansatz für Gesamtsicht
- Funktionalität durch abstrakte Namen besser ersichtlich

Motivation

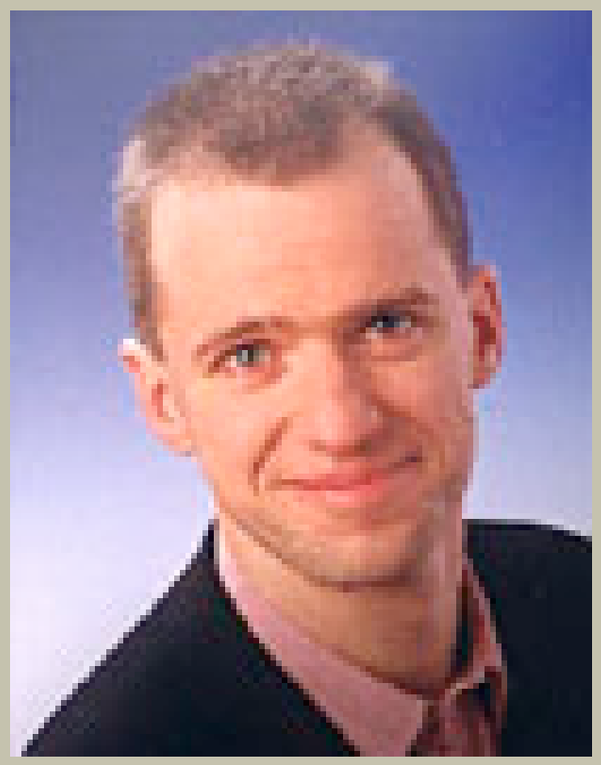
- Objektorientierung erschwert Kommunikation
- bisherige Untersuchengen dieses Problems nicht speziell für Entwurfsmuster
- Entwurfsmuster werden immer häufiger bei Namenskonventionen verwendet
- erstes formales Experiment zu diesem Thema

Motivation

- “Macht PCL Warungsarbeiten schneller?”
- “Macht man mit PCL weniger Fehler?”

Beschreibung

- Zwei ähnliche Experimente
  - Januar 1997 an der Universität Karlsruhe (UKA)
  - Mai 1997 an der Washington University St.Louis (WUSTL)



Lutz Prechelt

Barbara Unger-Lamprecht

Michael Philippsen

Walter F. Tichy

Experimententwurf

- Intra-Subjekt Entwurf mit Gegenbalancierung
- unabhängige Variable: PCL vorhanden oder nicht
- abhängige Variable:
  - Zeit die für die Wartung benötigt wird
  - Fehleranzahl
- zufällige Verteilung der Teilnehmer

Teilnehmer (UKA)

- 74 Teilnehmer, davon 64 Akademiker und 10 Studenten der Informatik
- 6-wöchiger Intensivkurs über Java und Entwurfsmuster
- durschnittliche Programmiererfahrung: 7,5 Jahre in 4,6 Programmiersprachen

Teilnehmer (UKA)

- größtes Programm: 3.510 LOC
- 69% haben Erfahrung mit objektorientierter Programmierung
- 58% haben Erfahrung mit GUI-Programmierung

Teilnehmer (WUSTL)

- 22 Teilnehmer, alles Informatikstudenten
- 12-wöchiger Kurs über C++ und Entwurfsmuster
- durschnittlichte Programmiererfahrung: 5 Jahre in 4,0 Programmiersprachen

Teilnehmer (WUSTL)

- größtes Programm: 2.557 LOC
- 76% haben Erfahrung mit objektorientierter Programmierung
- 50% haben Erfahrung mit GUI-Programmierung

Teilnehmer (Vergleich)

- UKA auf Experiment besser vorbereitet, da Vorbereitungskurs auf Experiment abgestimmt
- WUSTL nicht
- WUSTL: keine praktische Erfahrung mit Entwurfsmustern

Aufgabe

- zwei Programme bearbeiten
- Programme: gründlich kommentiert
- vier Gruppen:

1. erstes Programm mit PCL, dann zweites ohne PCL
2. erstes Programm ohne PCL, dann zweites mit PCL
3. zweites Prgramm mit PCL, dann erstes ohne PCL
4. zweites Programm ohne PCL, dann erstes mit PCL

Aufgabe (UKA)

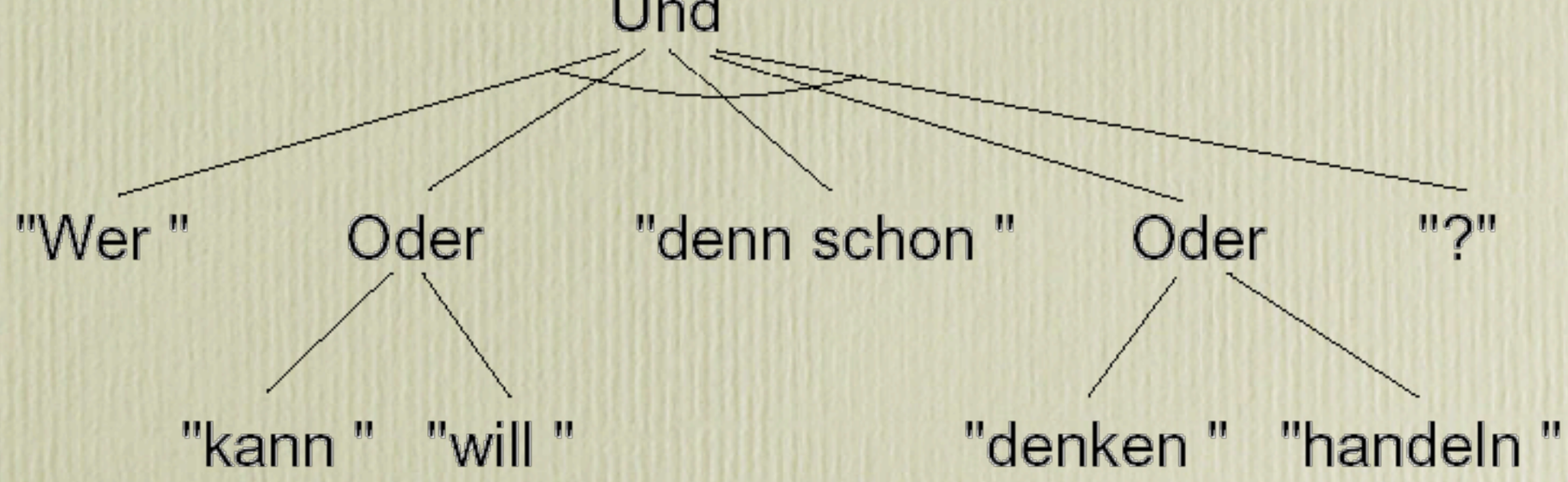
- Lösung auf Papier
- Programmiersprache: Java

Aufgabe (WUSTL)

- Lösung auf Unix Workstations
- Programmiersprache: C++

And/Or -Tree

- Anwendung zur Handhabung von Und/Oder Bäumen



And/Or -Tree

```
Wer kann denn schon denken ?
Wer kann denn schon handeln ?
Wer will denn schon denken ?
Wer will denn schon handeln ?
```

```
UND("Wer "&ODER("kann "|"will ")&"denn schon
"&ODER("denken "|"handeln ")&"?")
Tiefe=2 UndTiefe=0 OderTiefe=1
```

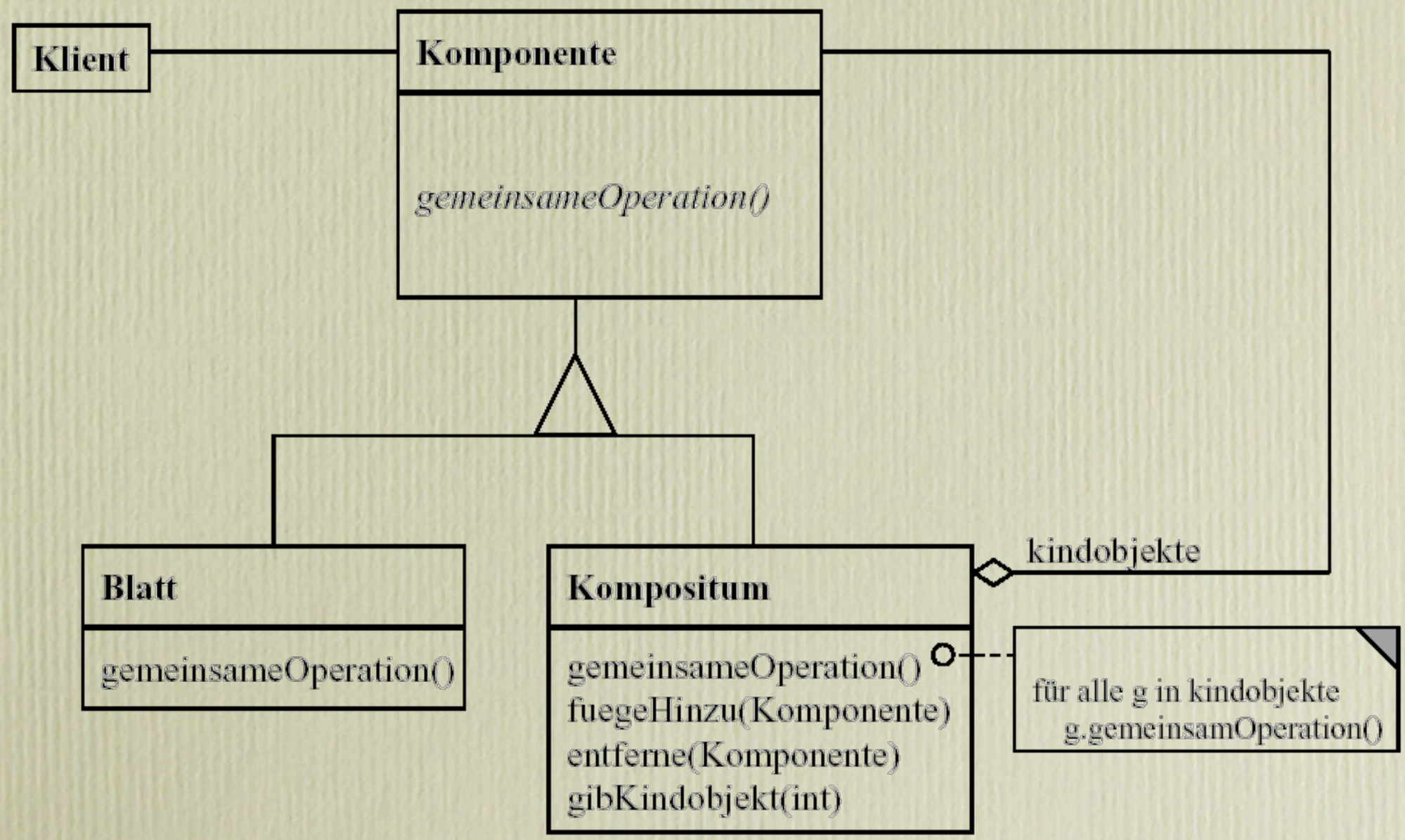
And/Or -Tree

- verwendete Entwurfsmuster:
  - Kompositum (Composite)
  - Besucher (Visitor)

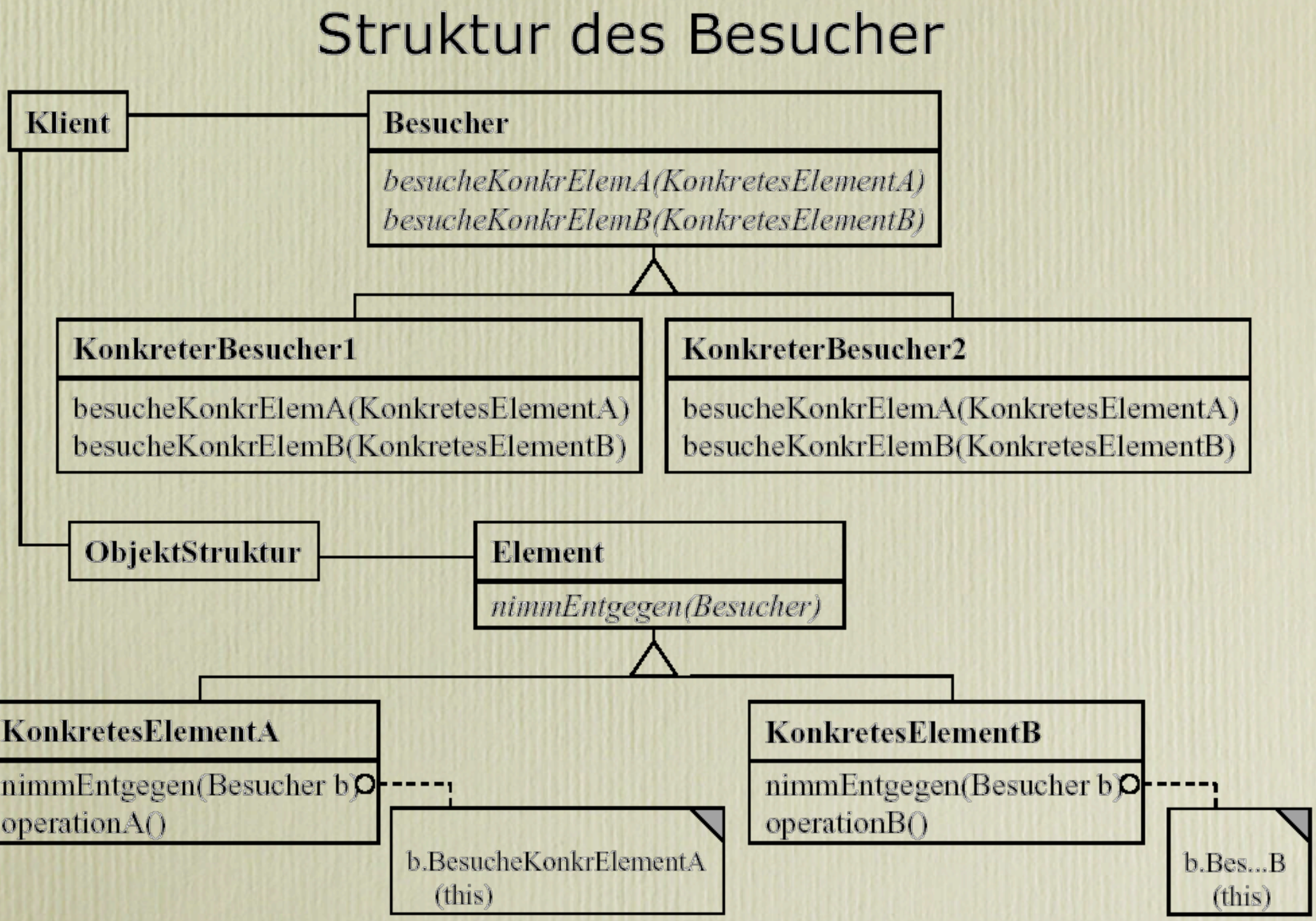
Exkurs: Kompositum

Struktur für Kompositum 2

Kompositum-Operationen im Kompositum



Exkurs: Besucher



And/Or -Tree

	UKA	WUSTL
LOC	362	498
Klassen	7	6
Kommentar	133	178
PCL	18	22

And/Or -Tree

```
//------------------------- Element -------------------------------
/**
  Schnittstelle der 'Element'-Klassen StringElement, AndElement, OrElement.
#M  *** ENTWURFSMUSTER: ***
#M  Element ist die Oberklasse in einem Kompositum-Muster.
*/
abstract class Element {
  /** Zufuegeoperation. Fuegt, falls moeglich, diesem Element ein weiteres
      Element zu.
  */
  abstract public void add(Element e);

  /** Repraesentationsoperation. Liefert eine kodierte Darstellung des
      Elements als String.
  */
  abstract public String asString();
```

And/Or -Tree

- Aufgaben

  erste Aufgabe: Finden der richtigen Stelle um Veränderungen der Ausgabe zu bewerkstelligen

  zweite Aufgabe: Formel angeben, wieviele Sätze sich aus dem gegebenen Syntaxbaum bilden lassen

And/Or -Tree

- Aufgaben

  dritte Aufgabe: Besucher-Klasse schreiben, die diese Anzahl schneller berechnen kann

  vierte Aufgabe: Geschriebene Klasse an der richtigen Stelle einbauen und Ergebnis ausgeben

Phonebook

- Programm stellt Eingaben des Benutzers (Name, Vorname, Telefonnummer) in verschiedenen Ansichten dar

- UKA: Java-Version mit GUI

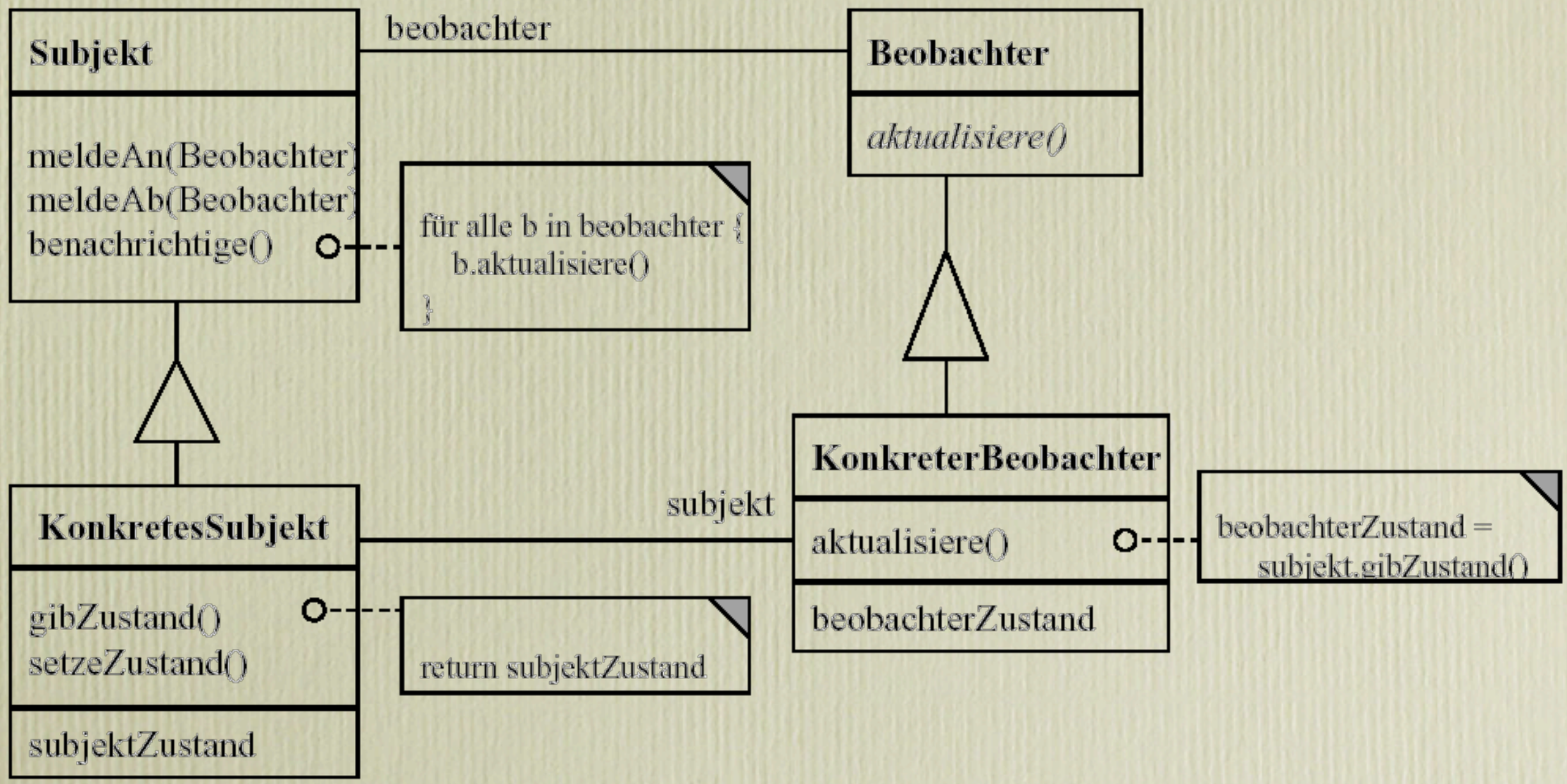
- WUSTL: C++-Version mit I/O-Stream

Phonebook

- verwendete Entwurfsmuster:
  - Beobachter (Observer)
  - Schablonenmethode (Template Method)

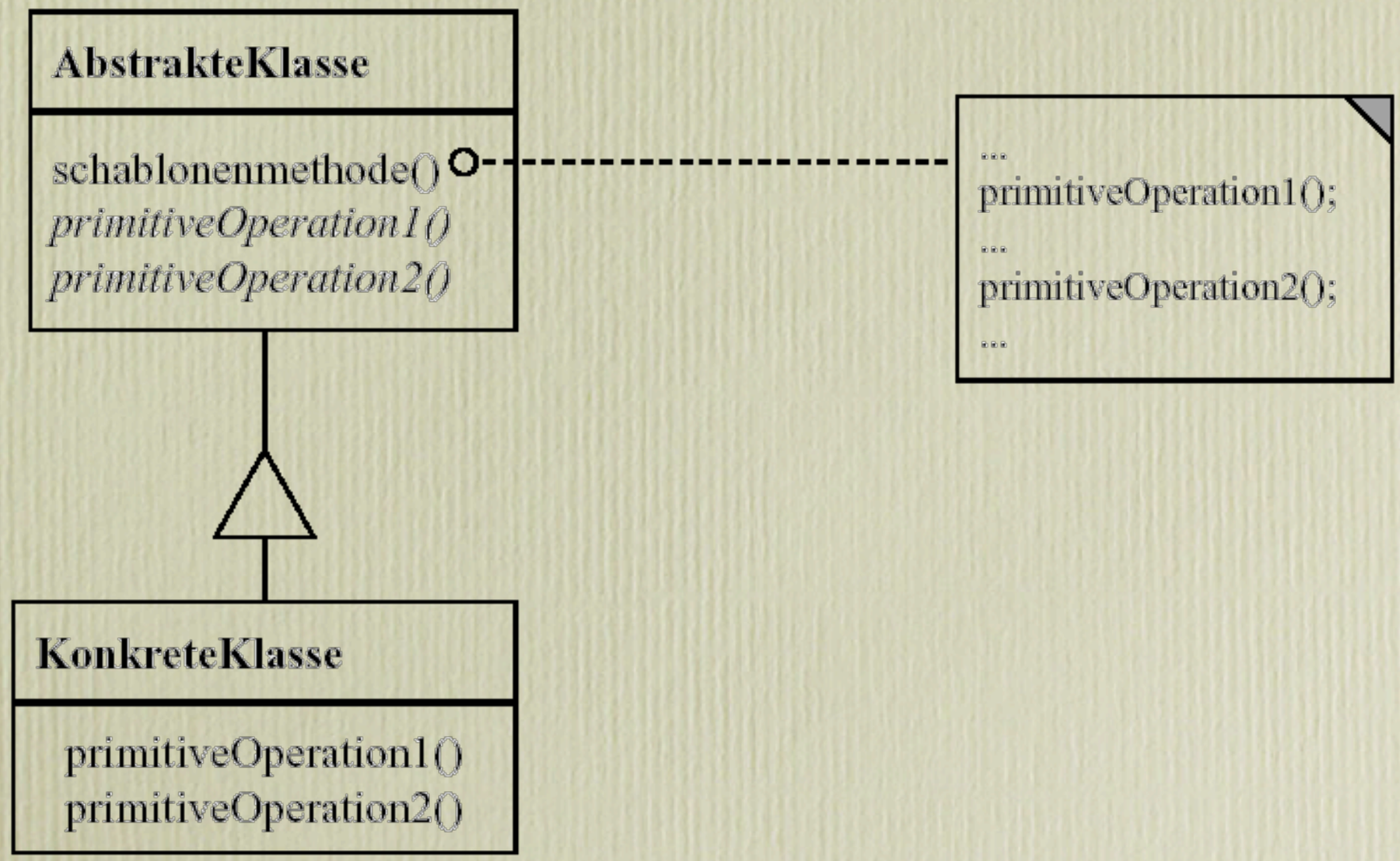
Exkurs: Beobachter

Struktur des Beobachters



Exkurs: Schablonenmethode

Struktur für Schablonenmethode



Phonebook

	UKA	WUSTL
LOC	565	448
Klassen	11	6
Kommentar	197	145
PCL	14	10

Phonebook

```
// Implements adding a new <Tuple>.  First <select> is used to test,
// whether the <Tuple> should be added at all, then <merge_in> moves
// it to the right place in the presentation using <less_than> and
// <format> converts it into a String of the desired display format.
//
// #M       *** DESIGN PATTERN: ***
// #M       new_tuple () together with its auxiliary method merge_in () forms a
// #M       **Template Method**. The empty spots that are filled in subclasses
// #M       are the methods select (), format (), and less_than ().

virtual void new_tuple (Tuple *name_number)
  {
    if (this->select (name_number) == 0)
      return; // This Tuple won't be displayed

    // Puts name_number into tuple_queue_ and tuple_string_vector_
    this->merge_in (name_number);
    this->display ();
  }
```

Phonebook

- Aufgaben

  erste Aufgabe: Finde eine Stelle für eine kleine Programmveränderung (Änderung der Formatierung der Ausgabe)

  zweite Aufgabe: Finde eine Stelle für eine kleine Programmveränderung (Änderung der Fenstergröße; nur UKA)

Phonebook

- Aufgaben

  dritte Aufgabe: Schreibe zusätzliche Beobachterklasse, welche die Schablonenmethode benutzt

  vierte Aufgabe: geschriebenen Beobachter einbauen

  fünfte Aufgabe: Schreibe weitere Beobachterklasse (keine Schablonenmethode)

Messungen

- Zeit
  - Ausgabe der Aufgaben bis zum Einsammeln
  - für jeden Teilnehmer separat
  - nicht für einzelne Teilaufgaben

Messungen

- Korrektheit

	And/Or -Tree		Phonebook	
	UKA	WUSTL	UKA	WUSTL
1.	2	2	2	2
2.	2	2	3	-
3.	8	8	8	8
4.	3	-	4	-
5.	-	-	6	8
insgesamt	15	12	23	18

Ergebnisse

- Hypothesen:

    1. Durch Hinzufügen von PCL werden Wartungsarbeiten schneller beendet
    2. Durch Hinzufügen von PCL werden bei Warungsarbeitern weniger Fehler gemacht.

Innere Gültigkeit

- Programmiererfahrung, Fähigkeiten, Lerneffekt, Motivation der Teilnehmer, äußere Bedingungen usw. beglichen durch zufällige Einteilung der Gruppen
- gegenbalancierter Entwurf kompensiert etwaige “unglückliche” Verteilung
- dominierende Störvariable: Sterblichkeit bei WUSTL
- zu wenige Teilnehmer um unstrittige Ergebnisse zu erhalten

Äußere Gültigkeit

- Erfahrung der Teilnehmer zu gering
  - erfahrenere Entwickler brauchen möglicherweise PCL nicht oder können es vielleicht effektiver nutzen
- kein Teamwork im Experiment
  - könnte Nutzen von PCL erhöhen, da bessere Kommunikation möglich

Äußere Gültigkeit

- Programmgröße und Komplexität
  - unrealistische Dimensionen
  - positive Effekte würden sich bei größeren Projekten verstärken
- Repräsentative Programme und Aufgaben
  - Verhältnis von Anzahl der Klassen zu Anzahl Entwurfsmuster ähnlich größeren Projekten

Äußere Gültigkeit

- Programme auch ohne PCL sehr gründlich dokumentiert
- in der Praxis meist nicht der Fall
- PCL eventuell hier noch hilfreicher

And/Or -Tree

- mehr “relevante” Punkte mit PCL
- Zeitbedarf mit PCL größer, aber dafür Anzahl der richtigen Lösungen ohne PCL deutlich niedriger
- Lösung auf Papier: Zeit für korrekte und nicht korrekte Lösungen nicht vergleichbar
    - Daher werden nur korrekte Lösungen verglichen: Zeitbedarf ähnlich

And/Or -Tree

	mit PCL	ohne PCL	Signifikanz
UKA			
relevante Punkte	8,5	7,8	0,20
korrekte Lsg.	15 von 38	7 von 36	0,077
Zeit	58,0	52,2	0,094
Zeit(korrekt)	52,3	45,4	0,17
WUSTL			
relevante Punkte	6,7	6,5	0,28
korrekte Lsg.	4 von 8	3 von 8	1
Zeit	52,1	67,5	0,046

And/Or -Tree

- PCL ist geringfügig langsamer, weil ...
    - die Gruppe größer ist,
    - die Teilnehmer möglicherweise weniger talentiert sind.

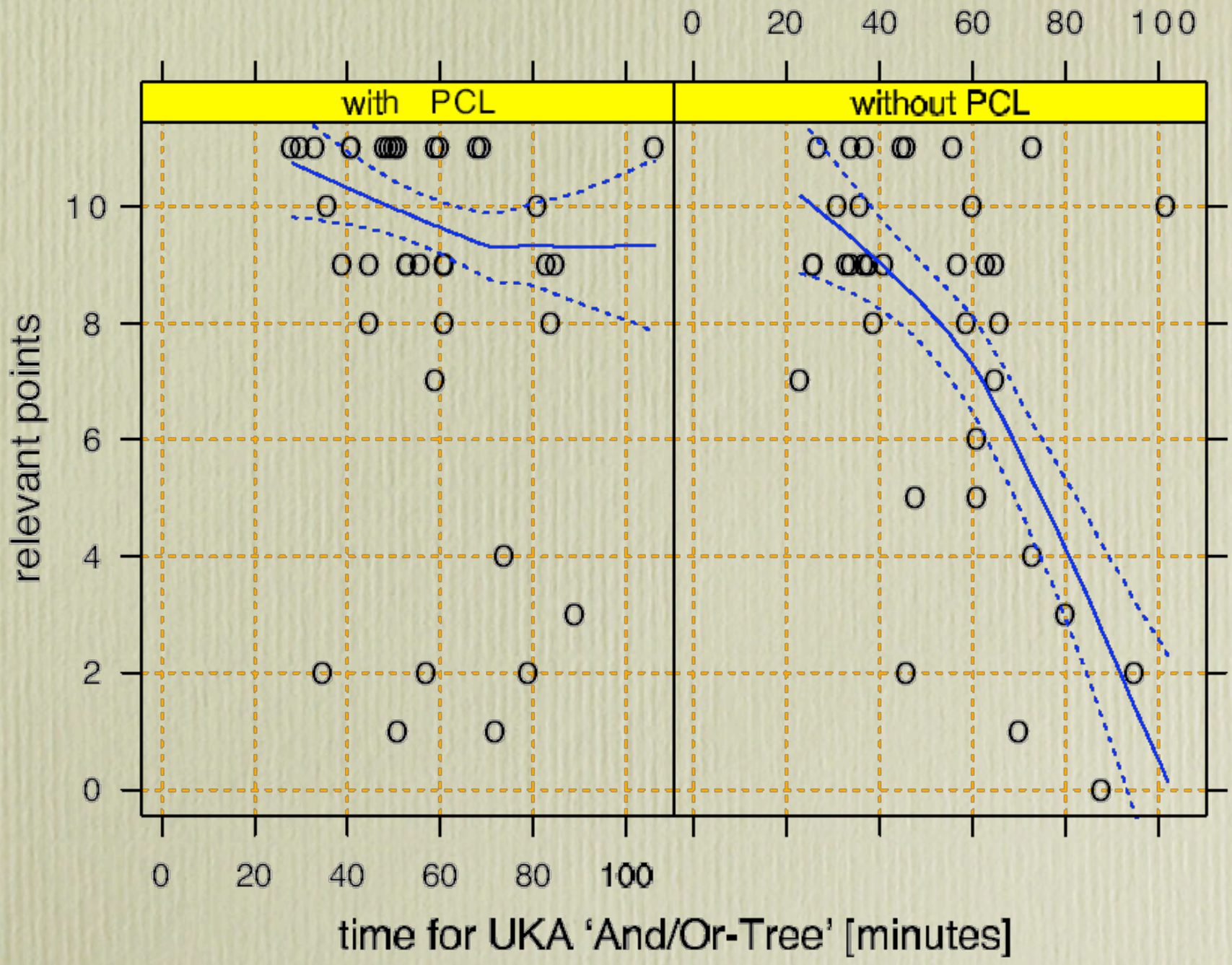
And/Or -Tree

- Speziell für WUSTL:
  - PCL schneller
  - Anzahl der richten Lösungen war gleich (Test durch Implementierung)
  - geringe Teilnehmeranzahl: man kann nicht nur die richtigen Lösungen betrachten (aber gleicher Trend)

And/Or -Tree

- Schlussfolgerung:
  - Trotz weniger talentierteren Teilnehmern ist die Anzahl der richtigen Lösungen höher.
  - Qualität der Lösungen mit PCL ist unabhängig von der benötigten Zeit.
  - 90% Wahrscheinlichkeit:
    PCL spart 0 bis 43% der Zeit für Wartungsarbeiten

And/Or -Tree


with PCL
without PCL
relevant points
time for UKA 'And/Or-Tree' [minutes]
0
20
40
60
80
100
0
2
4
6
8
10

And/Or -Tree

- Schlussfolgerung:
  - Für entwurfsmusterrelevante Wartungsarbeiten könnte PCL die Zeit und/oder die Fehler reduzieren.

Phonebook

- UKA:
  - Aufgabe war zu einfach
  - Ähnliche Ergebnisse wie bei And/Or-Tree
  - PCL schneller (um 0 bis 22%)
  - nur (fast) richtige Lösungen: PCL schneller

Phonebook

	mit PCL	ohne PCL	Signifikanz
UKA			
relevante Punkte	16,1	16,3	0,35
korrekte Lsg.	17 von 36	15 von 38	0,64
Zeit	51,5	57,9	0,055

Phonebook

- WUSTL:
  - Ergebnisse wertlos
    - nur 1 richtige Lösung (je Gruppe)
  - Aufgabe war zu schwierig
    (Beobachter ohne GUI, keine Erfahrung mit Beobachtern, keine Beispielklasse)

Ergebnisse

- Bei einfachen entwurfsmusterrelevanten Wartungsarbeiten könnte PCL Zeit sparen.

Ergebnisse

- Beide Hypothesen werden unterstützt, aber nicht bewiesen.
- Kein Ergebnis beweist beide Hypothesen gleichzeitig, aber es gibt auch keinen Gegenbeweis.
- In der Praxis wären die Auswirkungen von PCL möglichweise größer.

Bewertung

Experiment

- genereller Experimententwurf gut geeigent um Hypothesen zu beweisen
- Durchführung wies Mängel auf:
  - Lösung auf Papier
  - Interfaces bei Phonebook zu einfach
  - Sinn der nicht-relevanten Aufgaben

Experiment

- verlief bei WUSTL noch etwas unvorteilhafter
  - zu wenig Teilnehmer
  - ungünstiger Termin
  - schlechtes modifizieren der Aufgaben

Aufsatz

- gut strukturiert
- Analysen sind sinnvoll und gut begründet
- allerdings hätte kritischer auf die Durchführung eingegangen werden können
- These über weniger fähige Teilnehmer sehr fragwürdig

Zusammenfassung

- für wissenschaftlich formale und eindeutigere müsste eine erneute Untersuchung angestellt werden

- Fragestellung möglicherweise zu trivial

- Empfehlung sollte ausreichen, da Mehraufwand minimal und nutzen offensichtlich

DANKE !