

# Vererbung in der Softwarewartung

Aufbauend auf  
An Emprical Study Evaluating Depth of Inheritance on the  
Maintainability of Object-Oriented Software  
und  
A Controlled Experiment on Inheritance Depth  
as a Cost Factor for Code Maintanance

Georg Sisow

# „Eigenschaften“ der Vererbung

## Vererbung

- elegant
- leichtere und sicherere Wiederverwendung von Code
- weniger Redundanz
- reduziertes Defektpotenzial
- leichte Erweiterbarkeit im Zusammenhang mit Polymorphie
- einfaches, klares und flexibles Design

## Vererbung

- erzeugt zusätzliche Komplexität
- erzeugt Wechselwirkungen
- erzeugt Abhängigkeiten
- fördert Zerstreung

spekulative Bewertung

Aufgaben der Wartung werden von Verstehen dominiert!

Wie wirkt sich Vererbungstiefe darauf aus?

# Inhaltsübersicht

- I. Experiment Daly u.a.
  - I. Hypothesen
  - II. Aufbau
  - III. Durchführung
  - IV. Analyse
- II. Experiment Prechelt u.a.
  - I. Verbesserungen
  - II. Hypothesen & Aufbau
  - III. Durchführung
  - IV. Analyse
- III. Erklärungen und Modelle
  - I. Modelle
  - II. Erkenntnisse

# Aus Umfragen

Etwa die Hälfte der Befragten empfinden Vererbungstiefe als Verständnismfaktor

Ab Vererbungstiefe zwischen 4 und 6 beginnt die Vererbung erschwerend auf das Verständnis zu wirken.

# Experimententwurf

Untersuchung von Programmen der Vererbungstiefe 0, 3 und 5.

Forschungsmethode: Kontrolliertes Experiment

abhängige Variable: Zeit zur Erledigung der Aufgaben

unabhängige Variable: Vererbungstiefe

Rolle des kontrollierten Experiments: Hypothesentest

# Vorüberlegungen

H1 – Vererbung der Tiefe 3 beeinflusst nicht die Wartbarkeit von Softwaresystemen

H2 - Vererbung der Tiefe 3 beeinflusst die Wartbarkeit von Softwaresystemen.

# Vorüberlegungen

H3 – Vererbung der Tiefe 5 beeinflusst nicht die Wartbarkeit von Softwaresystemen

H4 - Vererbung der Tiefe 5 beeinflusst die Wartbarkeit von Softwaresystemen. Die Wartung erfordert mehr Zeit als die der „flachen“ Softwaresysteme.

# Experiment von Daly u.a.

## Aufbau Teil 1

- Teilnehmer: Absolventen eines Umschulungskurses
- Messung der Wartbarkeit in Zeitdifferenz
- gleiche Aufgabenstellungen für alle Programme
- nur eine Begrenzte Zeit zur Erledigung der Aufgaben
- Mehr-Faktor-Plan bzw. Gegenbalancierter Entwurf
- 31 Teilnehmer
- jeder Teilnehmer bekam bei jeder Aufgabe
  - ein Papier mit Versuchsanweisungen
  - ein Paket mit der Aufgabenstellung
  - ein Paket mit dem Quellcode

Kein Struktur  
verdeutlichendes  
Diagramm!



# Experiment von Daly u.a.

## Aufbau Teil 1

- Lösungen von einem Kontrollempfänger überprüft
- was in vorgegebener Zeit nicht fertig, wurde nicht berücksichtigt
- nach dem Experiment eine Nachbesprechungsfragebogen
- beide Programme einfache Datenbanken
  - Datenbank Universitätspersonal
  - Datenbank Schriftliche Arbeiten
- Programme ursprünglich in Vererbungstiefe 3 Versionen geschrieben
- „flache“ Versionen durch Einfügen von vererbten Code und Löschung der Oberklassen erstellt (etwas mehr Code dadurch in der flachen Versionen)
- Codegröße 360 bis 390 Zeilen

# Experiment von Daly u.a.

## Durchführung Teil 1

Gruppe A

4 Wochen objektorientierte Programmierung

Programm mit Vererbungshierarchie 3 erweitern. (Universitätspersonal)

Programm ohne Vererbung (flach) erweitern. (Schriftliche Arbeiten)

Gruppe B

Programm ohne Vererbung (flach) erweitern. (Universitätspersonal)

Programm mit Vererbungshierarchie 3 erweitern. (Schriftliche Arbeiten)

Aufgabenstellung:

gleich

Programm:

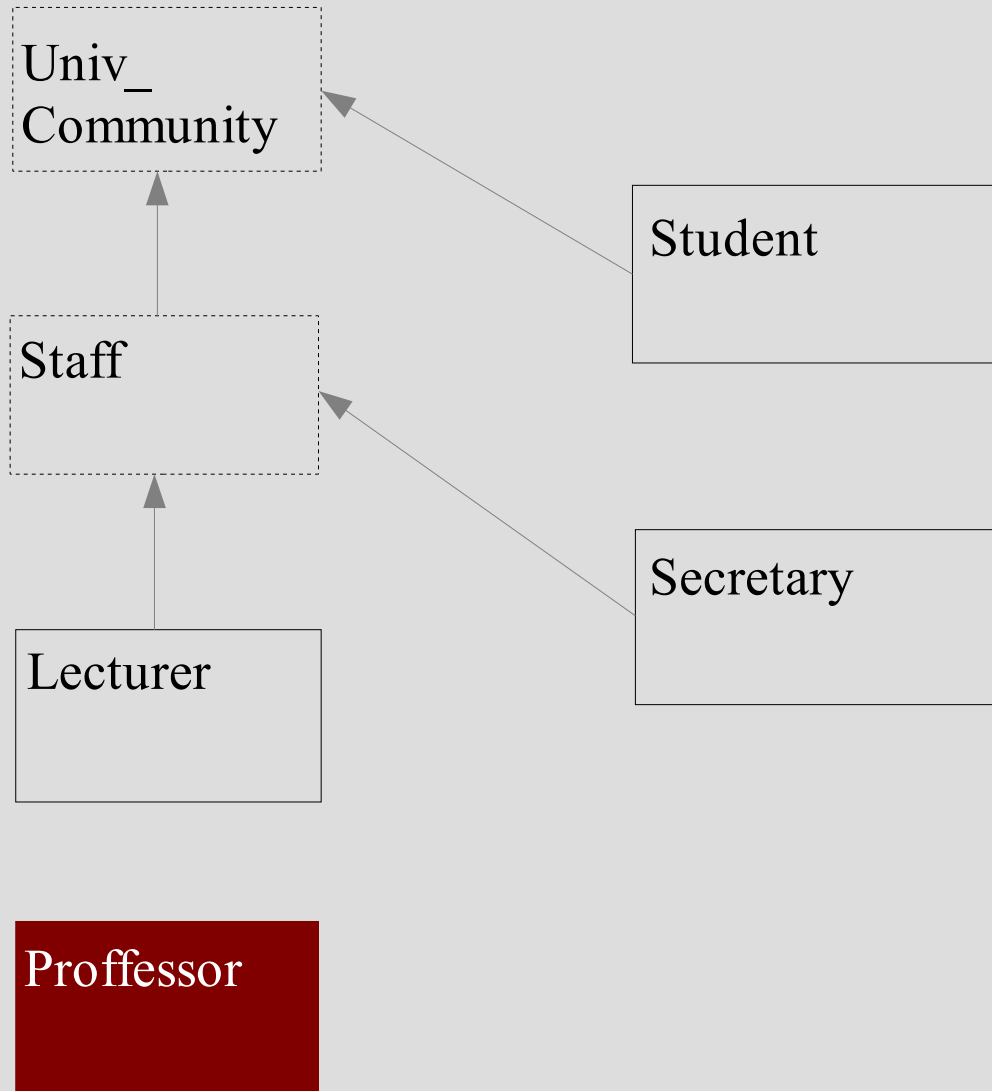
„gleich“

Struktur:

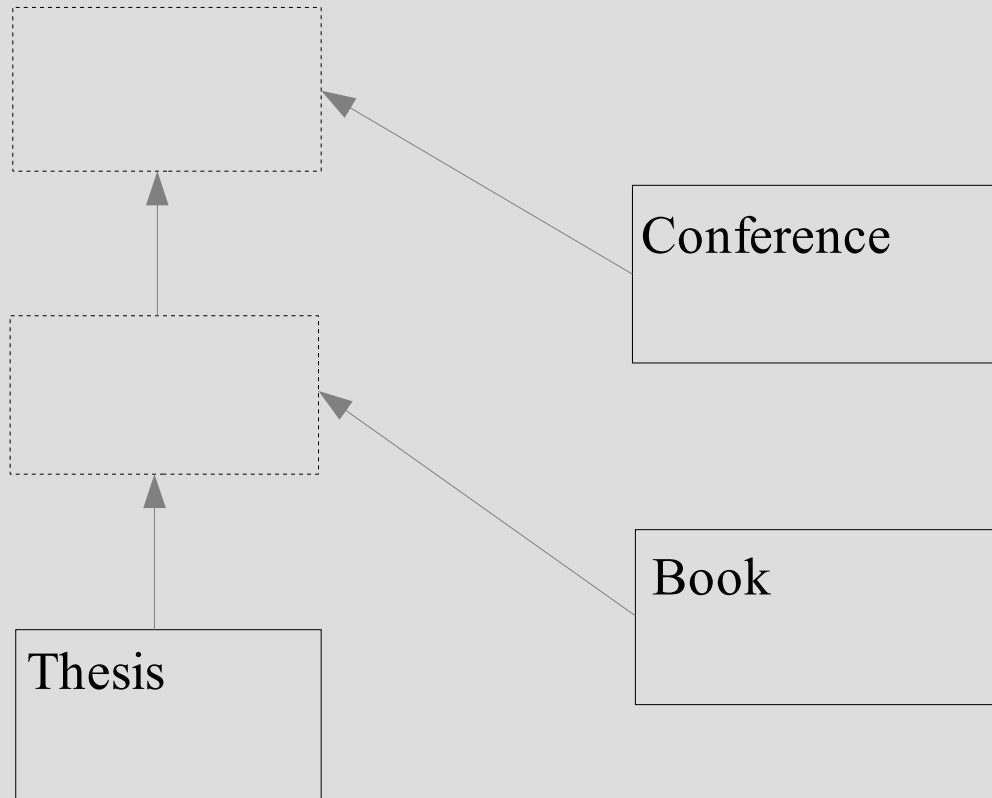
unterschiedlich

Spielt Struktur-reihenfolge eine Rolle?

# Vererbungshierarchie von Universitätsdatenbank



# Vererbungshierarchie von Schriftliche-Arbeiten-Datenbank



Phd\_Thesis

# Experiment 1

## Aufbau der Wiederholung (Wie Teil 1, aber ...)

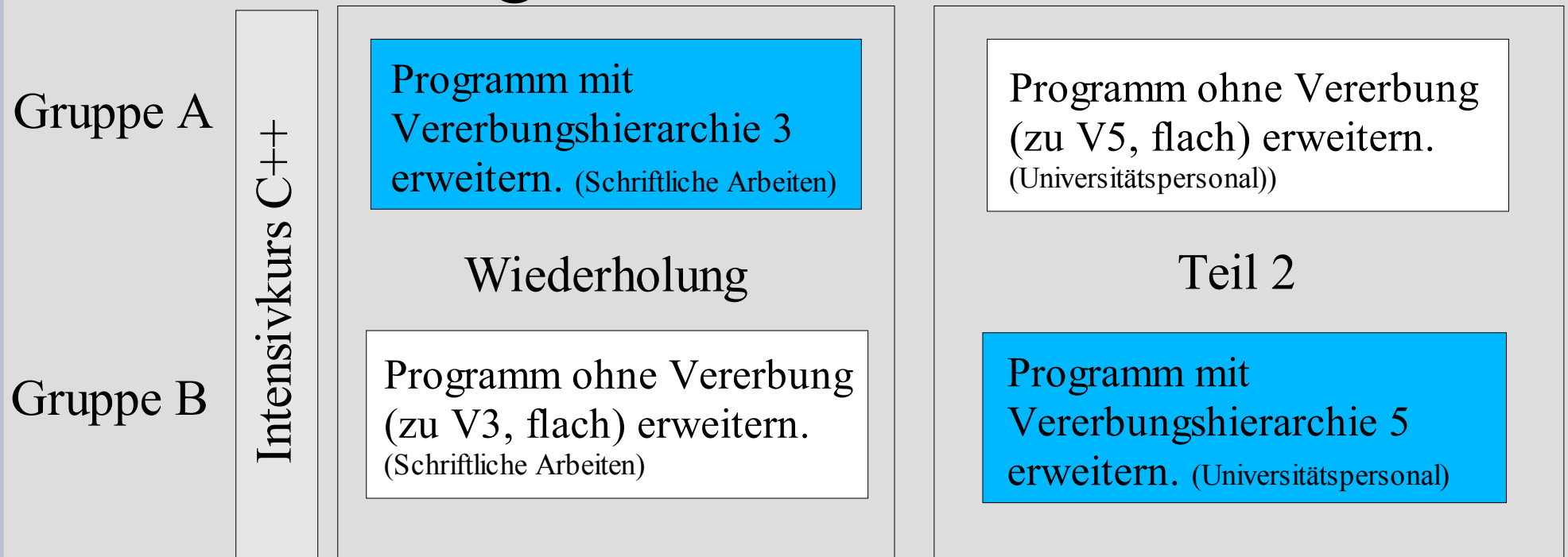
- Interne Wiederholung
- erfahrenere Teilnehmer aus kürzlichen Absolventen und kurz vor Abschluß stehenden Informatik Studierenden – ca 30 Personen
- Ausführung vor dem Bekanntwerden der Resultate des ersten Teils
- nur die Wartung der Schriftliche-Arbeiten-Datenbank wiederholt

## Aufbau des zweiten Teils (Wie Wiederholung, aber ... )

- eine größere Version der Universitätspersonal-Datenbank
- ca. 800 und 1100 Zeilen Code

# Experiment Daly u.a.

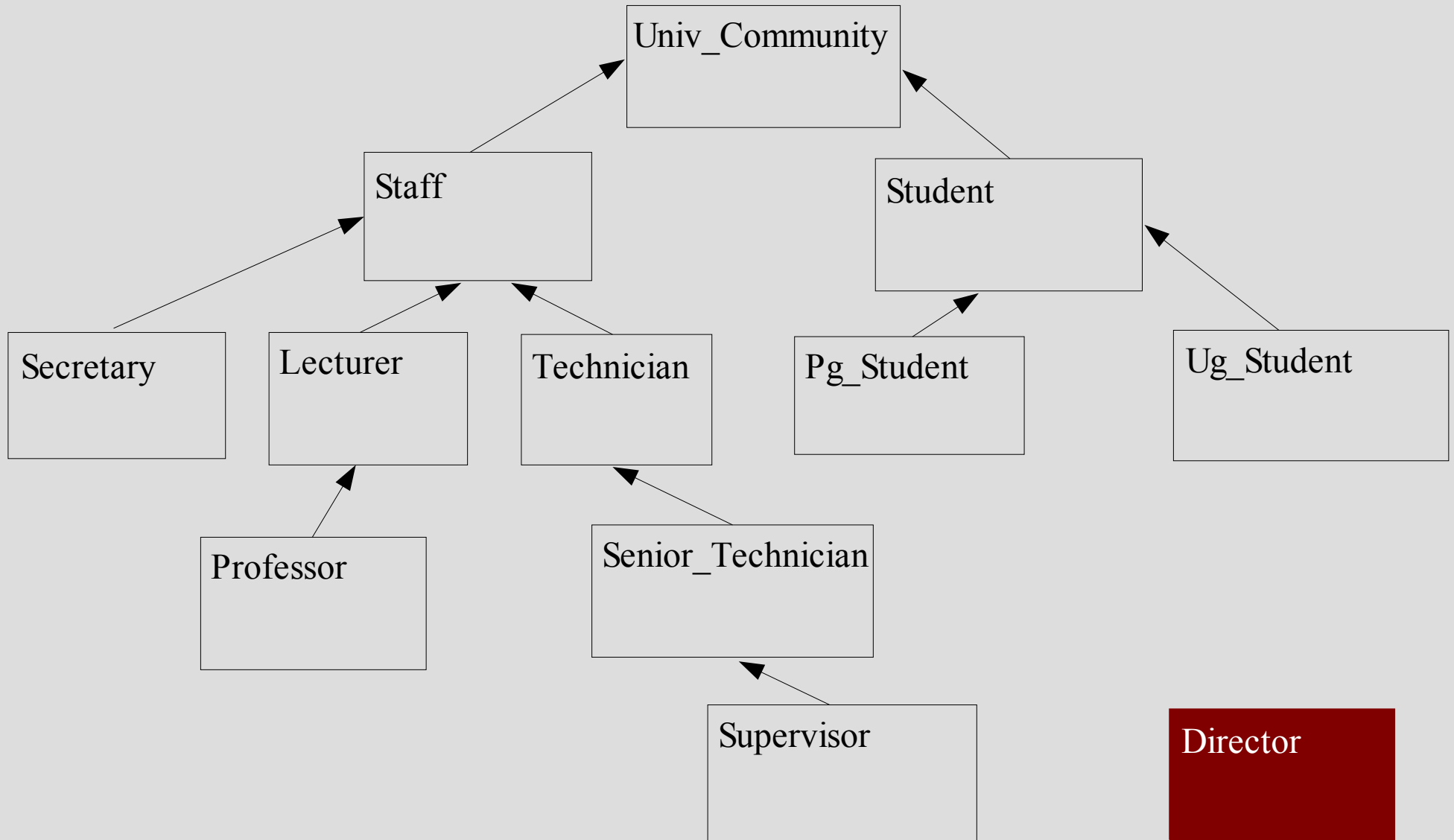
## Durchführung



Aufgabenstellung: gleich  
Programm: vergleichbar  
Struktur: unterschiedlich

Spielt Struktur-  
reihenfolge eine  
Rolle?

# Vererbungshierarchie von Schriftliche-Arbeiten-Datenbank



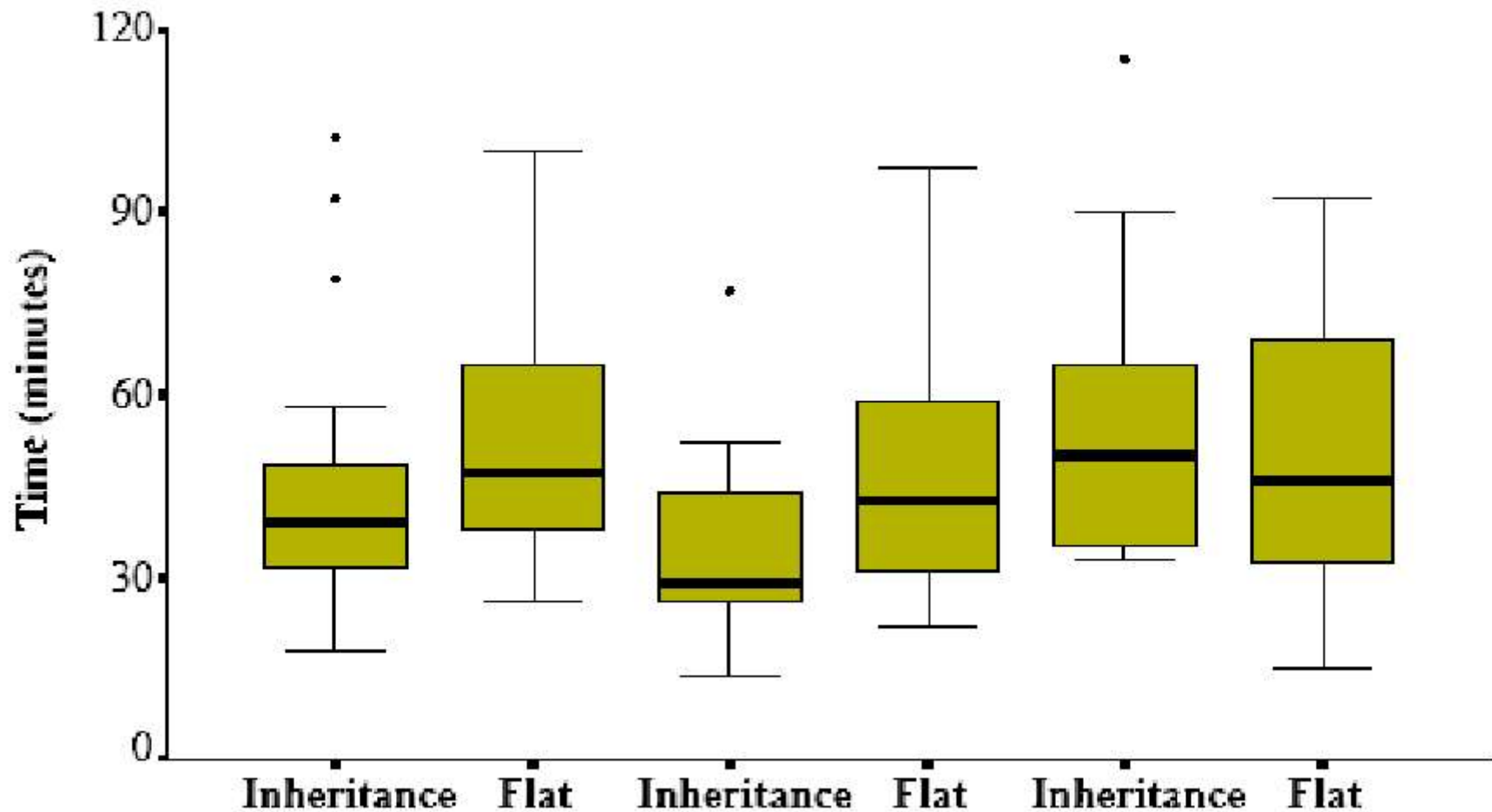
# Experiment Daly u.a.

## Gesammelten Daten

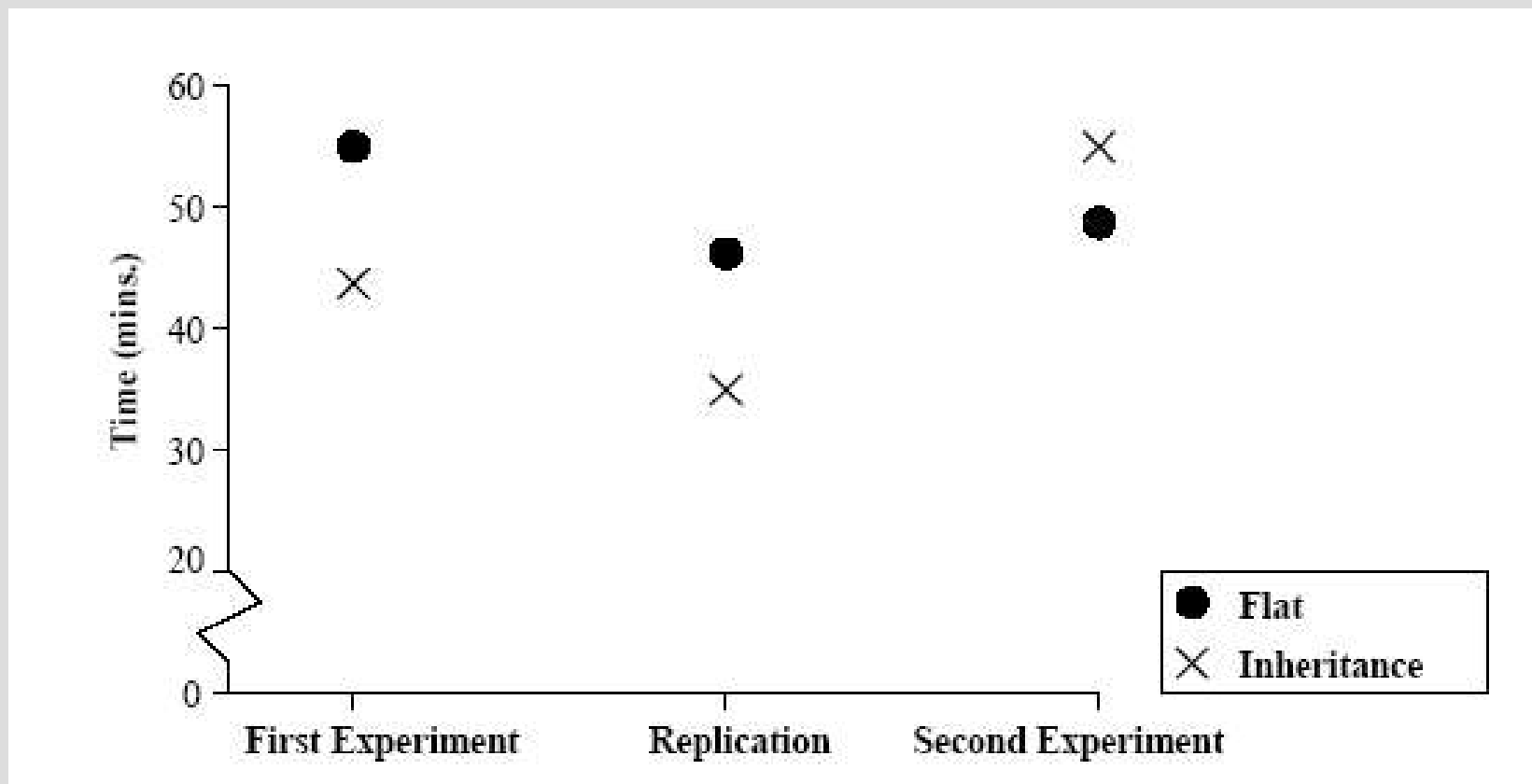
- (a) Zeit für die Fertigstellung der Aufgaben
- (b) Backups der Zwischenlösungen
- (c) Ein Skript über die Vorgehensweise
- (d) die letztendliche Lösung
- (e) Antworten auf den Nachbesprechungsfragebogen



# Experiment Daly u.a. - Auswertung



# Experiment Daly u.a. - Auswertung



# Experiment Daly u.a. - Auswertung

## Resultate

Die gesammelten Daten zeigen dass:

die Aufgaben an dem Softwaresystem mit Vererbungstiefe 3 schneller ausgeführt wurden als an dem mit Vererbungstiefe 0

die Aufgaben an dem Softwaresystem mit Vererbungstiefe 5 langsamer ausgeführt wurden als an dem mit Vererbungstiefe 0

die Nachträgliche Besprächung ergaben Anhaltspunkte das die Teilnehmer auf Schwierigkeiten in bei größeren Vererbungstiefe stiesen

# Experiment Daly u.a. - Analyse

## Analyse

### Innere Gültigkeit:

Auswahleffekte – wurden hier gut gelöst!

+ keine unterschiedliche Vorkenntnisse nötig

+ vorhergehende Ausbildung und Tests für alle gleich

Reifungseffekte – fast keine Anhaltspunkte dafür gefunden

+ unterschiedliche Programme

- gleiche Aufgaben

### Äusere Gültigkeit:

Repräsentativität der Teilnehmer - vertretbar

- keine Professionellen Softwareentwickler

+ Erfahrungsunterschied nicht so groß

Größe der Programme – nicht repräsentativ

+ die Vererbungsstruktur ist eine gute Skalierung

- Größe in LOC zu gering

# Experiment Daly u.a. - Analyse

## Störvariable: Lerneffekt

In Teil 1 sind im ersten Durchlauf 11 von 15 flache Lösungsansätze.

In zweiten Durchlauf sind 7 von 16 Lösungen flach.

—► es wurde eine gewisse „Sicherheit“ in Umgang mit Vererbung erlangt

Im Teil 2 (mit erfahrenen Programmierern) war kein Lerneffekt auszumachen.

## Randomisierung:

- Teilnehmer aufgrund von Testresultaten über ihr Wissen in objekt-orientierter Programmierung in Paare gruppiert
- anschließend jedes Paar in die Gruppe A und B aufgeteilt
- dadurch keine der Gruppen mit kompetenteren oder weniger kompetenteren Teilnehmern ausgestattet.

—► Vergleich einer Person mit sich selbst

# Experiment Daly u.a. - Analyse

Scheitern von relativ vielen in Teil 1 (11 die mindestens bei einer Aufgabe nicht erfolgreich waren) - Vermutung auf Überforderung der Gruppe

eingeschränkte Vergleichbarkeit der 3er Versionen mit der 5er Version - begründet durch Konzentration auf Aufgaberelevanten Programmteil

Verdopplung der Klassen in Teil 2 führen zu einem Suchproblem für die Teilnehmer (dabei spielt das fehlende konzeptuelle Modell eine wichtige Rolle)

Eine mögliche Interpretation der Resultate ist das Teil 1 und Wiederholung die Vorzüge der Vererbung offenbahrt, während im Teil 2 die Vorzüge der Wiederholung durch Suchprobleme verschüttet wurden.

# Experiment Daly u.a. - Analyse

	<i>Probleme eine Oberklasse/Schablone zu wählen</i>	<i>Probleme bei der Nachverfolgung der Vererbungshierarchie</i>	<i>Probleme mit vererbten Funktionen</i>	<i>Probleme mit dem Fehlen des konzeptuellen Modells</i>	<i>Einschlagen von einer optimalen Strategie</i>	<i>Trotz erkannter üblicher Strategie, eine weniger als optimale gewählt</i>	<i>Einschlagen einer befriedigenden Strategie</i>	<i>Klasse zur Ableitung in weniger als 5 min gewählt</i>	<i>Klasse zur Ableitung in 5 bis 10 min gewählt</i>
<b>Flach</b>	1	0	1	1	<b>6</b>	0	1	<b>4</b>	1
<b>Vererbung</b>	4	5	4	6	<b>7</b>	3	1	1	<b>7</b>

# Experiment Prechelt u.a.

## Replikation der Ergebnisse

1. Die Struktur des Vererbungsbaums in Form eines Klassendiagramms wird den Teilnehmern zur Verfügung gestellt.
2. Programme sind nicht so einfach: mehrere Arten von Beziehungen, nicht offensichtliche Funktionalität
3. Verwendung des gleichen Programms in allen Versionen der Vererbungstiefe.
4. Schwerere und vorallem unterschiedliche Wartungsaufgaben.
5. Kein Kontrolleur bei Abgabe der Lösung.
6. Java als Programmiersprache



# Vorüberlegungen

H5 – Programme mit größerer Vererbungstiefe brauchen mehr Zeit in der Wartung.

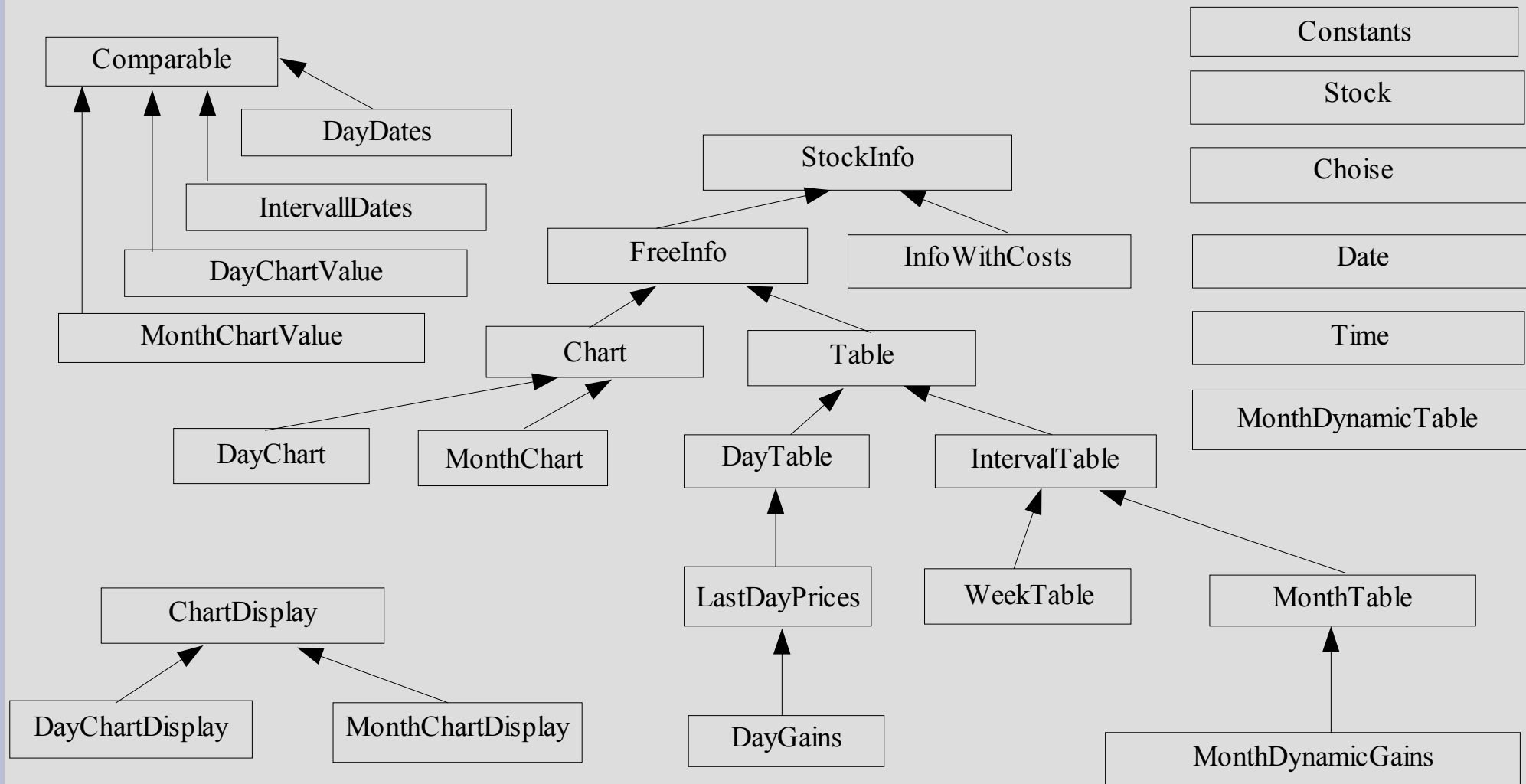
H6 – Programme mit größerer Vererbungstiefe sind Ursache schlechterer Qualität in der Wartung.

# Experiment Prechelt u.a.

## Aufbau

- Teilnehmer: Gruppe G - 57 Absolventen in Informatik  
Gruppe U - 58 Studenten im ersten Jahr
- flachere Programmversionen wurden aus dem ursprünglichen Vererbungstiefe 5 Programm erstellt
- Programmgröße zw. 1200 und 2500 LOC
- matched-between-subject design
- als abhängige Variable dazu: die Qualität
- jeder Teilnehmer beendete die Bearbeitung nach eigenem Ermessen
- Aufgaben in denen es um das Verstehen geht
- Vorhergehende Lehre: ein 6 Wochen intensiv Java Kurs bei Absolventen  
Veranstaltung „Informatik 2“ bei Anfängern
- Programmiererfahrung: ca. 8 Jahre bei Absolventen  
ca. 6 Jahre bei Anfängern

# Experiment Prechelt u.a.



# Experiment Prechelt u.a.

## Durchführung

Gruppe G

**Aufgabe 1**

A1: ?, A2: ?, A3: ?

**Aufgabe 2a + 2b**

A1: ?, A2: ?, A3: ?

Gruppe U

**Aufgabe 2a**

B1: ?, B2: ?, B3: ?

**Aufgabe 2b**

B1: ?, B2: ?, B3: ?

Fragen zum Hintergrund-  
wissen & Vortest

Nachbesprechungsfragebogen

Aufgabenstellung: gleich  
Programm: gleich  
Struktur: unterschiedlich

# Experiment Prechelt u.a.

- Aufgabe 1: solving the Year-2000-Problem  
ganze Programm auf 4 stellige Jahresdatum umstellen  
Vererbungstiefe 5 Programm mit geringsten Änderungen
- Aufgabe 2a: neue Art von Anzeige implementieren  
Verhalten am meisten in der Vererbungstiefe 5 Programm  
verstreut + braucht ein gutes Verständniss der Beziehungen in  
der Vererbung im Programm
- Aufgabe 2b: eine andere neue Art von Anzeige implementieren  
kann in Vererbungstiefe 5 Programm leicht gelöst werden denn  
es kann die Lösung von 2a mit nur einer Änderung verwenden  
+ braucht ein gutes Verständnis der Beziehungen in der  
Vererbung des Programms

# Experiment Prechelt u.a.

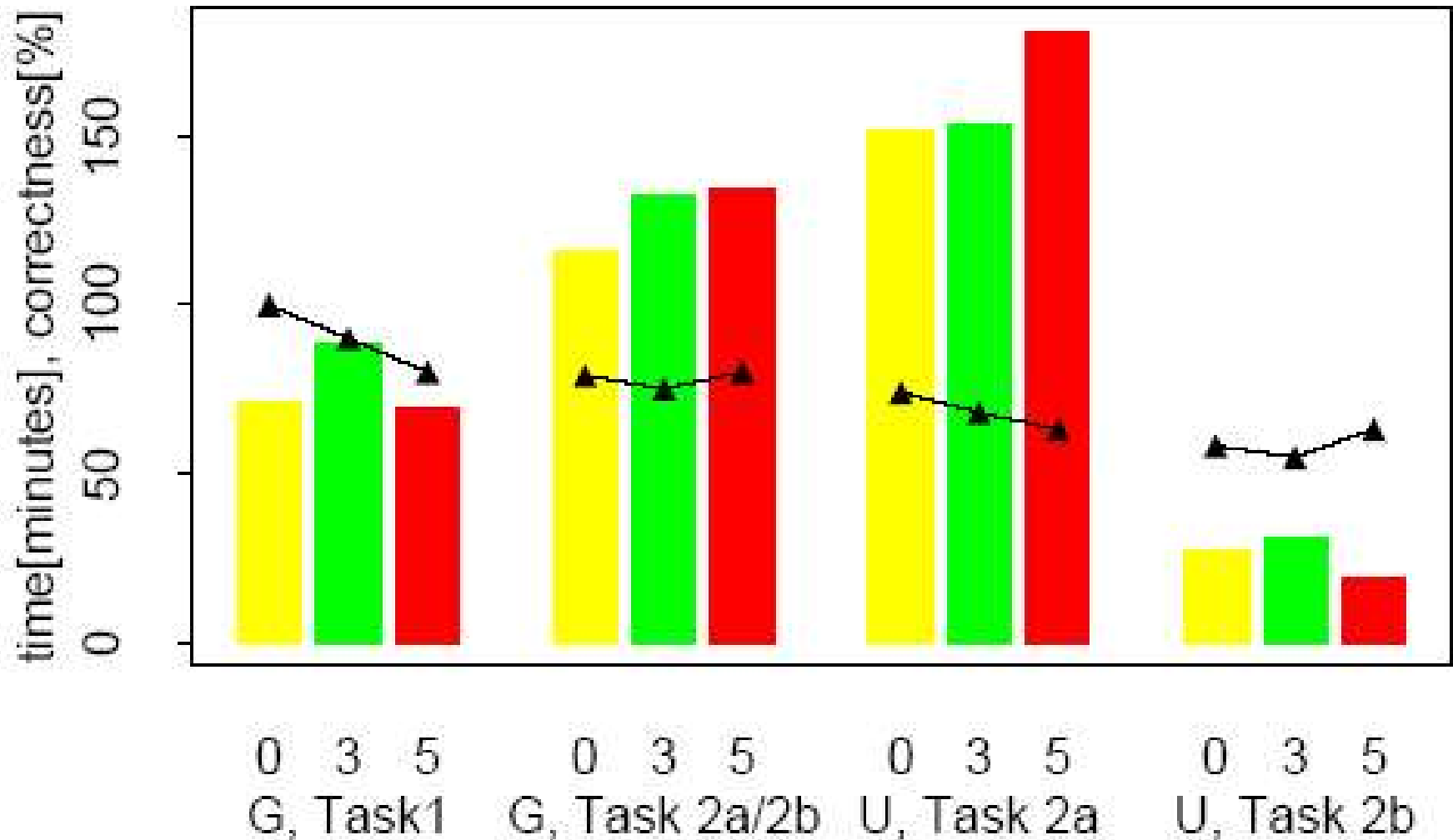
## Gesammelte Daten

- Hintergrundwissen Umfrageergebnisse und Vortestergebnisse
- Zeit für jede Aufgabe
- jede Lösungsstufe und Testdurchlauf
- Antworten auf den Nachbesprechungsfragebogen

## Datenvorverarbeitung

Die Lösung der Aufgaben wurde in Bezug auf die Erfüllung der Aufgabenstellung eingestuft.

# Experiment Prechelt u.a.



# Experiment Prechelt u.a. - Analyse

## Innere Gültigkeit:

- verschiedene Programme durch Umwandlung
- unterschiedliche Gruppen durch Zufall

## Äußere Gültigkeit:

- die Teilnahme von Anfängern
- Vertrautheit mit dem Programm in der realen Umgebung
- andere Dokumentation und Werkzeuge in der realen Umgebung
- Art der Wartungsaufgaben



# Experiment Prechelt u.a. - Analyse

## Fragen an Teilnehmer

- Verwendung der Vererbung —————▶ *zu wenig in der flachen Version*
- Korrekte Lösung —————▶ *am korrektesten empfand es die Gruppe mit der flachen Version*
- Klarheit der Programmstruktur —————▶ *Absolventen die V 5 Version die Anfänger die anderen beiden*
- Schwierigkeit der Aufgaben —————▶ *flache Version am einfachsten*
- Konzentration —————▶ *schwerer je mehr Vererbungstiefe*

Mit steigender Vererbungstiefe werden Verstehen und Warten von Programmen sowohl langsamer als auch fehleranfälliger.

# Andere Experimente

Wiederholung des Vergleichs von Vererbungstiefe 3 Softwaresystemen mit äquivalenten „flachen“ Versionen. [Cartwright, 1998]  
Resultat wies die flachen Programme als signifikant schneller wartbar aus.

Wiederholung des Vergleichs aller Vererbungstiefen 0, 3 und 5 Softwaresysteme in einem rein schriftlichen Experiment, wo es allein auf die Korrektheit der Lösungen ankam. [Harrison u. a., 1999]  
Resultat: die Vererbungstiefe 3 und 5 Softwaresysteme provozierten mehr Fehler als ihre flachen äquivalenten.

# Erklärungen und Modelle

Verwendung der Daten aller Experimente!

Mögliche Variablen zur Eingabe:

**Programmeigenschaften:**

Vererbungstiefe, Anzahl Klassen,  
Anzahl Methoden, Codezeilen, ...

**Aufgabeneigenschaften**

Anzahl zu verstehender Methoden, Anzahl  
neuer oder geänderter Klassen, Methoden,...,  
nötige Veränderungen der Hierarchie

**Teilnehmereigenschaften**

ungefähre Einstufung der Erfahrung

# Erklärungen und Modelle

Modelle zur Vorhersage der Gruppendurchschnittszeiten:

A) $t = f(\text{Vererbungstiefe})$	10% der Variabilität	!!!
B) $t = f(\text{Hierrarchiesprünge})$	55% der Variabilität	
C) $t = f(\text{Methoden zu verstehen})$	84% der Variabilität	
D) $t = f(\text{Vererbungstiefe, Methoden zu verstehen})$	85% der Variabilität	!!!
E) $t = f(\text{Methoden zu verstehen, Erfahrung})$	91% der Variabilität	
F) $t = f(\text{Vererbungstiefe, Methoden zu verstehen, Erfahrung})$	92% der Variabilität	!!!
G) $t = f(\text{Methoden zu verstehen, Erfahrung})$	94% der Variabilität	
H) $t = f(\text{Vererbungstiefe, Methoden zu verstehen, Erfahrung})$	96% der Variabilität	

Die Vererbungstiefe ist als Vorhersagefaktor für die Wartbarkeit untauglich!

# Methoden - der Schlüssel?

Estauulich gute Vorhersagen mit „Methoden zu verstehen“.

Dennoch: Nur erklärbar für Aufgaben die von Verstehen beherrscht werden.  
Dies kaum im Voraus zu ermitteln wie viele Methoden nötig sind.

Die Wartung wird von Verstehen dominiert!

Es ist aber der technische Aspekt der hier aus den Augen gerät.

Die Suche nach Modellen für Vorhersagen zum Aufwand der Wartung sollte Eigenschaften die in Beziehung zur Vererbungstiefe stehen mehr zur Rate ziehen als die Vererbungstiefe selbst.

Bei strategischen Entscheidungen wo Wartungsaufgaben bekannt sind kann mit dem Methodenfaktor eine gute Hilfe zur Verfügung stehen.

# Viel erreicht?

## „Resultate“

Die Annahmen in den Versuchen waren Irreführend.

Die Vererbungstiefe ist nicht der Faktor der den Wartungsaufwand bestimmt!

## „Gewinne“

Ein Modell das mit einer Variablen immerhin 70% der Varianz in den Arbeitszeiten der Experimente erklärt.

Für weitere Erforschung beeinflussender Faktoren müssen eher die begleitenden Faktoren der Vererbungstiefe untersucht werden.

Ende