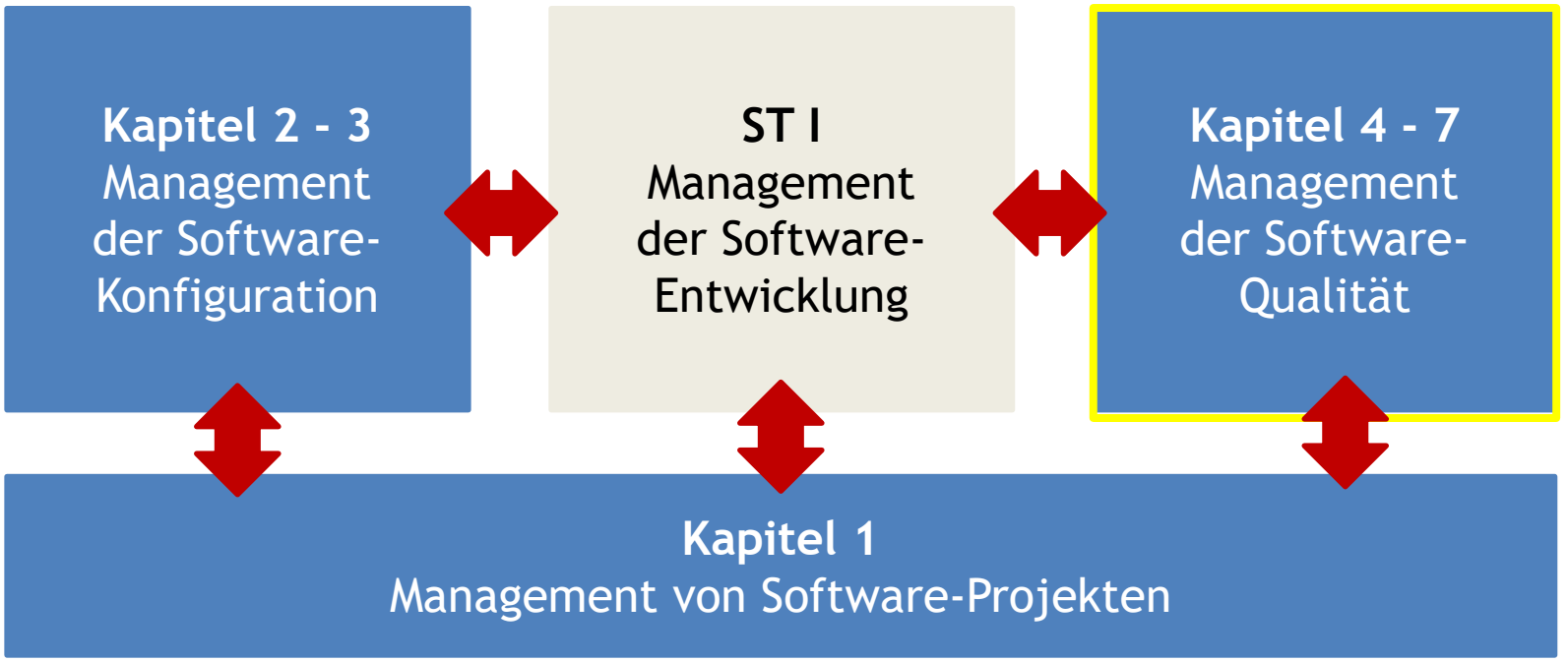
UNIVERSITÄT SIEGEN

# Vorlesung Softwaretechnik II

## Management der Software-Qualität: Empirische Analysen

# Aufbau der Vorlesung



# Inhalt

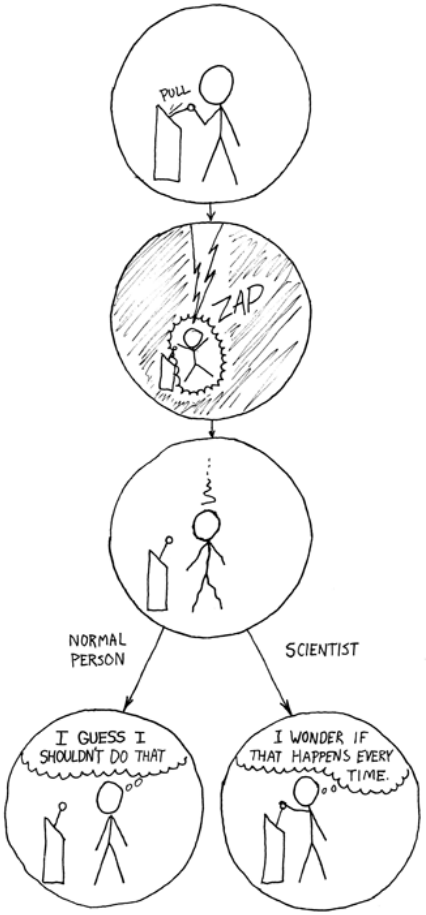
- Einführung und Motivation
- Planung, Durchführung und Dokumentation von experimentellen Evaluationen

UNIVERSITÄT SIEGEN

# Einführung und Motivation

# Forschung vs. Praxis

The scientist builds
in order to study;
the engineer studies
in order to build.




[F. Brooks 1996]

# Zitat

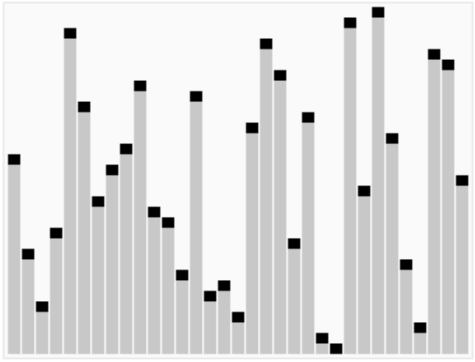
*When we write programs that "learn", it turns out that we do and they don’t.*

[Perlisisms - "Epigrams in Programming" by Alan J. Perlis]

# Theorie vs. Praxis

- Theorie: „unangreifbare“ Beweise formaler Eigenschaften mathematischer Objekte (z.B. Quelltexte von Programmen)
- Praxis: „subjektive“ Wahrnehmung beobachtbarer Eigenschaften „realer“ Objekte (z.B. Programmausführungen auf einer Hardware-Plattform)

# Beispiel: Komplexität von Sortieralgorithmen



- **QuickSort**
  - Best Case: O(n log n)
  - Average Case: O(n log n)
  - Worst Case: O(n^2)
    (Pivot ist immer min/max Element der Liste)
- **BubbleSort**
  - Best Case: O(n) (sortierte Liste)
  - Average Case: O(n^2)
  - Worst Case: O(n^2) (umgekehrt sortierte Liste)

# Beispiel: Komplexität von Algorithmen

- Komplexitätstheorie als Auswahlhilfe für einen „geeigneten“ (d.h. effizienten) Sortieralgorithmus.
- *Schlussfolgerung*: QuickSort ist besser als BubbleSort!?

# Beispiel: Theorie vs. Praxis

- *Beobachtung*: QuickSort schneidet im Average Case und Worst Case besser ab als BubbleSort, aber BubbleSort schneidet im Best Case besser ab als QuickSort.
- *Schlussfolgerung*: Die Eignung des gewählten Sortieralgorithmus ist abhängig von Verteilungseigenschaften (z.B. Häufigkeit der Worst Cases) in den **zu erwartenden Eingabedaten (Faktor „Anwendungskontext“).**

# Beispiel: Theorie vs. Praxis

- *Beobachtung*: Rekursive / parallelisierte Algorithmen (z.B. QuickSort) sind für den Entwickler/Anwender schwieriger zu verstehen und zu debuggen, als iterative / sequentielle Algorithmen (z.B. BubbleSort).
- *Schlussfolgerung*: Die Eignung des gewählten Sortieralgorithmus ist abhängig vom **Grad der Akzeptanz des Entwicklers/Anwenders (Faktor „Mensch“).**

# Grenzen der Theorie

Probleme theoretischer Untersuchungen:

- Aufwand (Experten, Zeit, Ressourcen, …)
- Validität (Modell bildet Realität korrekt ab, Beweise nachvollziehbar, …)
- Grenzen (Modellbildung nicht möglich, keine Theorie bekannt, Unentscheidbarkeit von Eigenschaften, …)

# Grenzen der Praxis

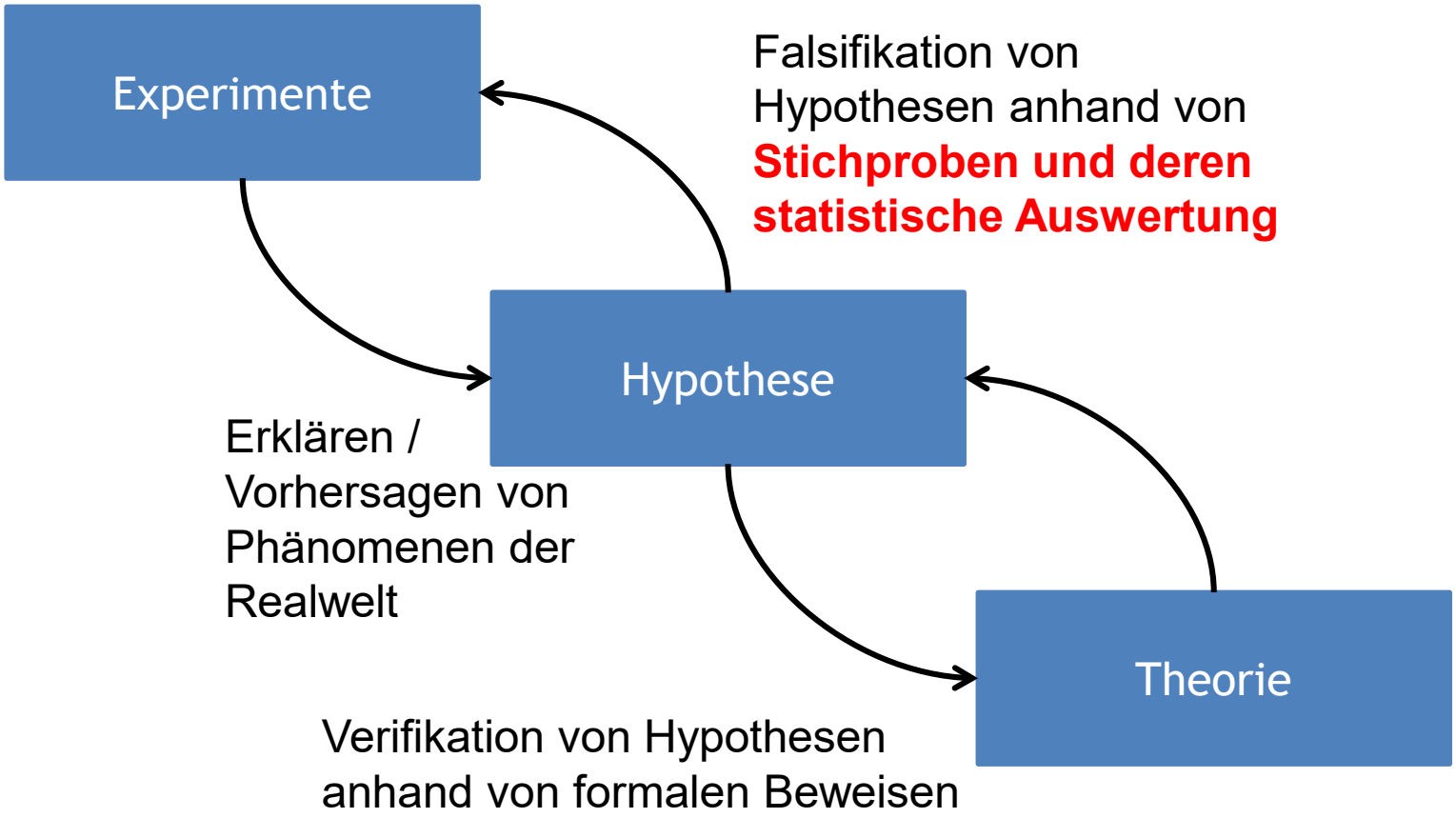
Probleme praktischer Untersuchungen:

- Betrachtung **aller möglichen Anwendungskontexte** (z.B. alle möglichen Eingabelisten) vor der Inbetriebnahme unmöglich.
- Einbeziehung **aller möglichen menschlichen Faktoren** (z.B. alle möglichen Arten von Anwendern) vor der Inbetriebnahme unmöglich.

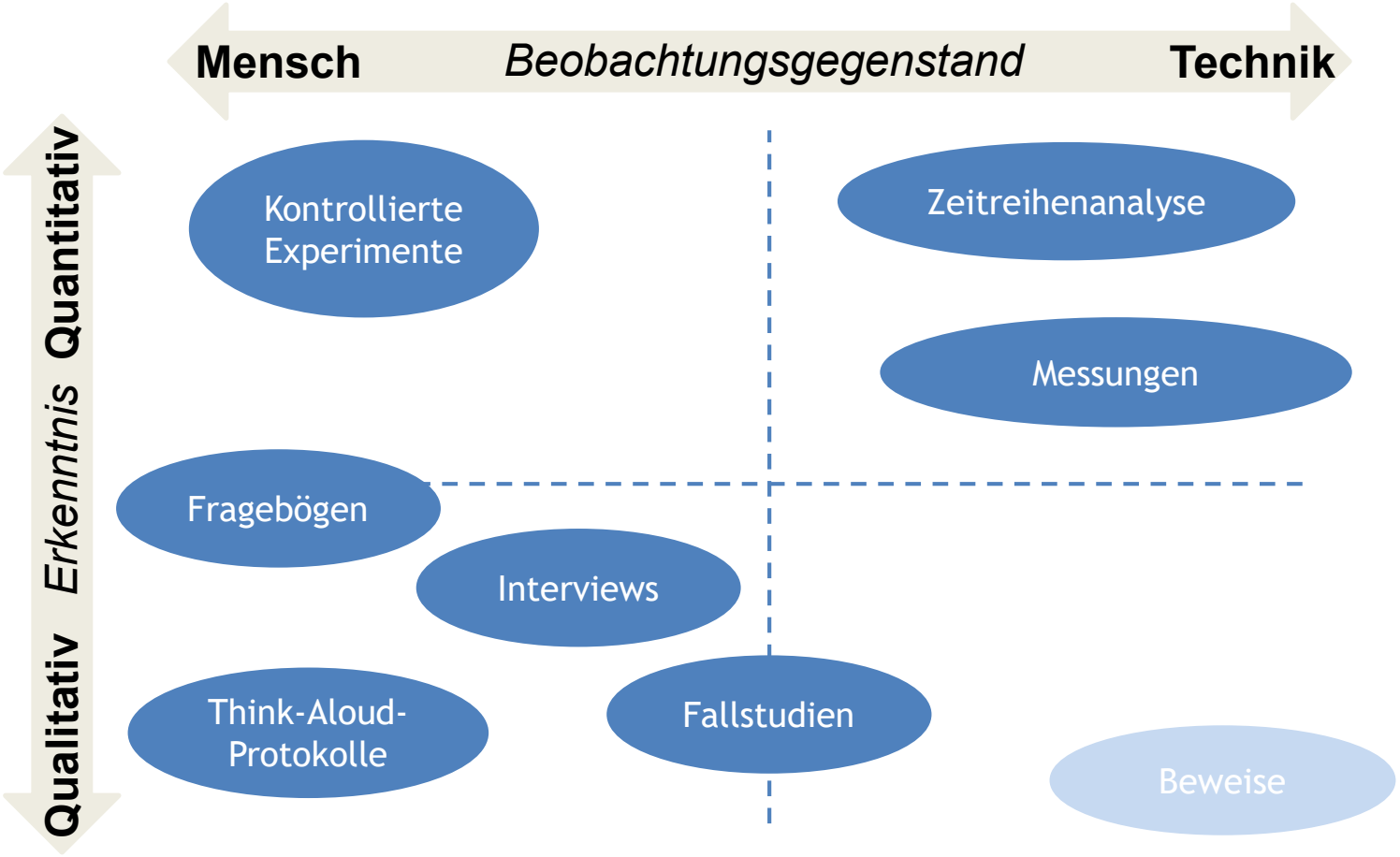
# Empirie

- Griechisch (empeiría): Erfahrung, Beobachtung
- Duden:
  a) Methode, die sich auf wissenschaftliche Erfahrung stützt, um Erkenntnisse zu gewinnen
  b) aus wissenschaftlicher Erfahrung gewonnenes Wissen; Erfahrungswissen

# Die wissenschaftliche Methode



# Empirische Methoden



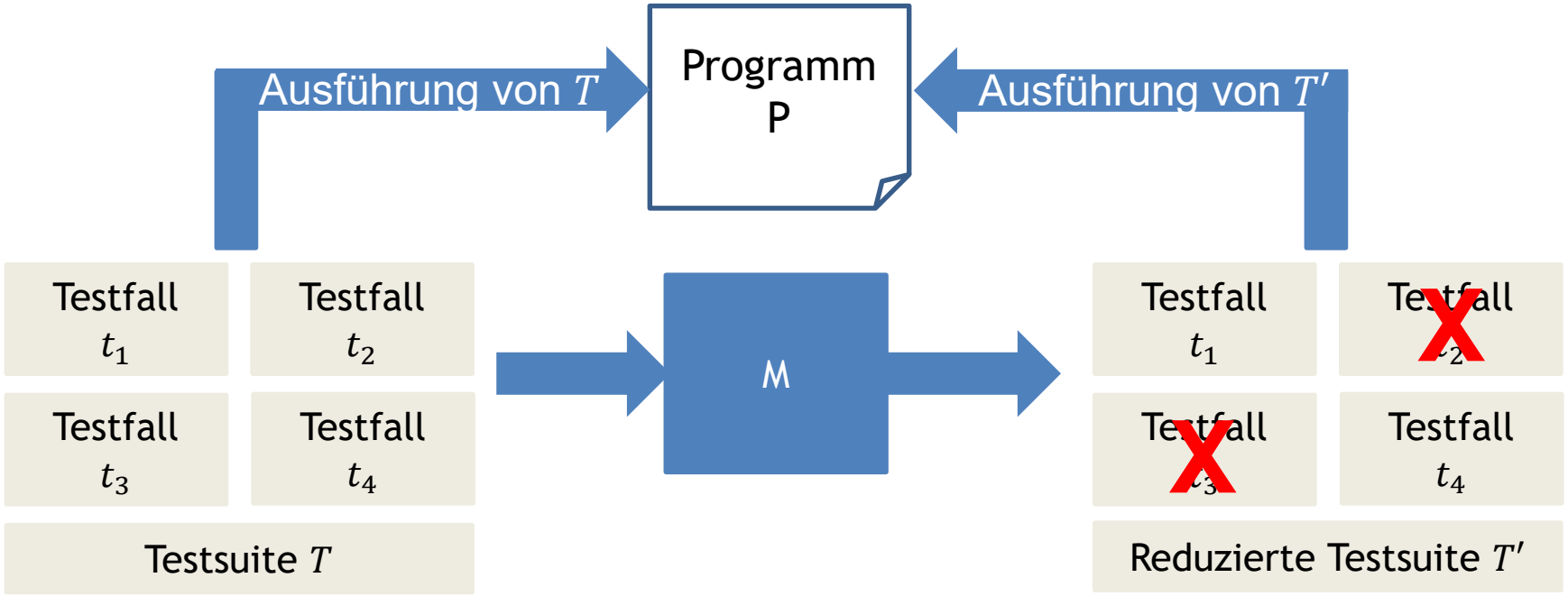
# Beispiele für empirische Resultate

- *„Durch Pair Programming wird die Anzahl von Programmierfehlern reduziert, falls die Partner unterschiedliche Skill-Level aufweisen.“*
- *„Durch Copy & Paste erstellter Code versucht mehr Programmfehler, als neu geschriebener Code“*
- …

UNIVERSITÄT SIEGEN

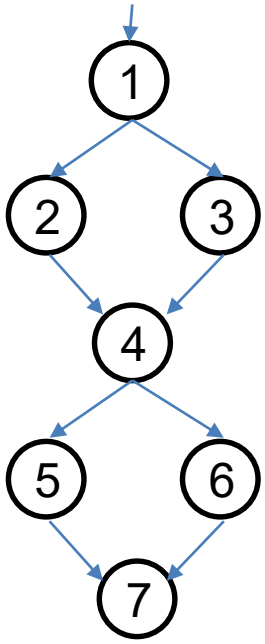
# Planung, Durchführung und Dokumentation von experimentellen Evaluationen

# Beispiel: Testsuite-Reduktion




Es wird eine neue Methode „M“ zur Reduktion von bestehenden Testsuiten entwickelt, um Programme P schneller (mit weniger Testfällen, aber gleicher Abdeckung) zu testen.

# Beispiel: Testsuite-Reduktion




Kontrollflussgraph von Programm P

Testfall
$t_1$ = 1-2-4-5-7

Testfall
$t_2$ = 1-2-4-6-7

Testfall
$t_3$ = 1-3-4-5-7

Testfall
$t_4$ = 1-3-4-6-7

Testsuite $T$ (Zweigüberdeckung)

M

Testfall
$t_1$ = 1-2-4-5-7

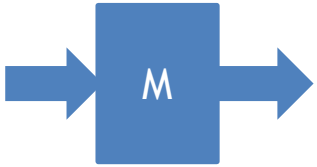
Testfall
$t_2$ = 1-2-4-6-7 X

Testfall
$t_3$ = 1-3-4-5-7 X

Testfall
$t_4$ = 1-3-4-6-7

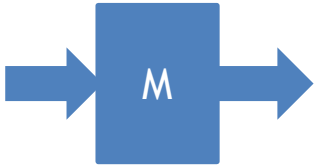
Reduzierte Testsuite $T'$ (Zweigüberdeckung)

# Beispiel: Testsuite-Reduktion




- Es soll experimentell untersucht werden, ob Methode „M“ den Software-Test in der Praxis verbessern kann.
- Zur Erinnerung: verschiedene Testmethoden können anhand der **Effizienz** (Testaufwand) und **Effektivität** (Testerfolg) bewertet werden.

# Beispiel: Testsuite-Reduktion



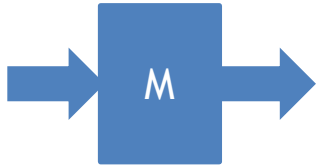

Konkrete Fragestellungen:

- Effizienz: Wird der zusätzliche Aufwand zur Reduktion der Testsuite durch die reduzierte Testdauer amortisiert?
- Effektivität: Wird durch die Entfernung von Testfällen aus einer Testsuite, die redundant hinsichtlich eines Abdeckungskriteriums sind, möglicherweise doch die Fehlerdetektionsrate der Testsuite negativ beeinflusst?

# Experimentelle Evaluation: Zielsetzung

- Im ersten Schritt wird die Zielsetzung der Evaluation anhand einer/mehrerer **Hypothesen** oder **Forschungsfragen** formuliert.
- Hypothesen sind zumeist konkreter (strikter) formuliert und führen zu einem eindeutigem ja/nein-Ergebnis.
- Forschungsfragen sind hingegeben offener formuliert und erlauben differenziertere / detailliertere Antworten.

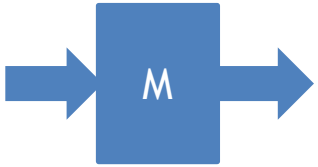
# Beispiel: Hypothesen




*H1: Erhöht die Anwendung von M die Testeffizienz?*
*(Wunschergebnis: ja)*

*H2: Reduziert die Anwendung von M die Testeffektivität?*
*(Wunschergebnis: nein)*

# Beispiel: Forschungsfragen




*F1: Welchen Einfluss hat die Anwendung von M auf die Testeffizienz?*
*(Wunschergebnis: Erhöhung um X % - mit möglichst großem X)*

*F2: Welchen Einfluss hat die Anwendung von M auf die Testeffizienz?*
*(Wunschergebnis: Keinen, bzw. Verringerung um Y % - mit möglichst kleinem Y)*

# Design der Experimente

- Welche Experimente müssen durchgeführt werden und welche Daten dabei erhoben / gemessen werden, um die Hypothesen zu belegen / widerlegen bzw. die Forschungsfragen zu beantworten?
- Identifikation von **Faktoren**, die Auswirkungen auf die Messergebnisse haben können.
- Festlegung von **Experimentparametern** und dem **Korpus**.

# Beispiel: Experimentparameter

- Welche Testmethodik? *Entscheidung*:
  (Ausschließlich Funktionstests auf Unit-Level)
- Welche Abdeckungskriterien? *Entscheidung*:
  (C0, C1)
- Wie werden ursprüngliche Testsuiten ausgewählt?
  *Entscheidung*:
  (Existierend, Zufällig, Automatischer Testfallgenerator)
- Welche Fehlerarten zur Bewertung der Effektivität?
  *Entscheidung*:
  (Reale Fehler, Mutanten)

# Beispiel: Experimentparameter

- Ergebnis: 4 Experimentparameter
- Da Abhängigkeiten zwischen Experiment-parametern Auswirkungen auf Messergebnisse haben können, müssen Messungen für sämtliche Parameter-Kombinationen wiederholt durchgeführt werden.
- Somit: 12 Messreihen pro Eingabeprogramm P:

*(Funkt.+Unit)* × *(C0, C1)* × *(Ex., Zuf., Gen.)* × *(Real., Mut.)*

# Beispiel: Auswahl eines Korpus

Bei der Auswahl von Eingabeprogrammen (**Subject Systems**) in den Experiment-Korpus ist zu prüfen:

- Verwendete Programmiersprache einheitlich und verarbeitbar durch den Testfallgenerator und das Mutationswerkzeug?
- Programm-Units separat testbar?
- Dokumentation realer Fehler verfügbar?
- Existierende ursprüngliche Testsuite für die betrachteten Abdeckungskriterien verfügbar?

# Beispiel: Auswahl eines Korpus

Ist der Korpus „adäquat“?

- Anzahl der Programme ausreichend?
- Umfang und Komplexität der gewählten Programme repräsentativ?
- …

Manchmal gibt es etablierte Benchmarks (bewährte Sammlungen von Beispiel-Programmen speziell für bestimmte Fragestellungen)

# Beispiel: Auswahl eines Korpus

| Subject | Function Name | # Variants | # LOC | # Faults |
|---|---|---|---|---|
| BusyBox | du_main | 4 | 123.5 | 16 |
| | id_main | 4 | 237.5 | 24 |
| | readlink_main | 4 | 62 | 4 |
| | sleep_main | 3 | 130.66 | 14 |
| | tail_main | 4 | 536.25 | 82 |
| | wc_main | 4 | 315.5 | 24 |
| VerCS | email_test | 16 | 117.5 | 4 |
| | minepump_test | 4 | 67.25 | 1 |

- Beispiel: Darstellung relevanter Informationen des gewählten Korpus in Tabellenform.
- Zusätzlich: Größe der ursprünglichen Testsuite, Anzahl realer Fehler, Anzahl Mutanten, …

# Sammlung der Daten: Metriken

- Auswahl und Beschreibung der erhobenen **Messgrößen** mit der verwendeten **Metrik**.
- Beschreibung der **Aggregation** oder Art der **Mittelwertbildung** für zusammengesetzte Ergebnisse.
- Gruppierung nach Hypothesen / Forschungsfragen.

# Beispiel: Sammlung der Daten

**F1 (Effizienz)**

- *Messgröße* „Ursprüngliche Testsuite-Größe“:
  Anzahl der Testfälle in der Original-Testsuite für alle Programme des Korpus und alle Abdeckungskriterien.
  *Metrik*: Positive Ganzzahl.
- *Messgröße* „Testsuite-Größe nach Reduktion“:
  Anzahl der Testfälle in der reduzierten Testsuite für alle Programme des Korpus und alle Abdeckungskriterien.
  *Metrik*: Positive Ganzzahl.

# Beispiel: Sammlung der Daten

**F1 (Effizienz)**

- *Messgröße* „Ursprüngliche Testdauer“:
  Dauer der Ausführung aller Testfälle der Original-Testsuite für alle Programme des Korpus und alle Experimentparametrisierungen.
  *Metrik*: CPU-Zeit in Sekunden / Speicherbedarf in MB
- *Messgröße* „Testdauer nach Reduktion“:
  Dauer der Ausführung aller Testfälle der reduzierten Testsuiten für alle Programmen des Korpus und alle Experimentparametrisierungen + Dauer der Testsuite-Reduktion
  *Metrik*: CPU-Zeit in Sekunden / Speicherbedarf in MB

# Beispiel: Sammlung der Daten

**F2 (Effektivität)**

- *Messgröße* „Ursprüngliche Fehlerdetektionsrate“:
  Anteil der durch Testfälle in der Original-Testsuite entdeckten Fehler für alle Programme des Korpus und alle Experimentparametrisierungen.
  *Metrik*: Prozent.
- *Messgröße* „ Fehlerdetektionsrate nach Reduktion“:
  Anteil der durch Testfälle in der reduzierten Testsuite entdeckten Fehler für alle Programme des Korpus und alle Experimentparametrisierungen.
  *Metrik*: Prozent.

(Gleiches für Mutationen)

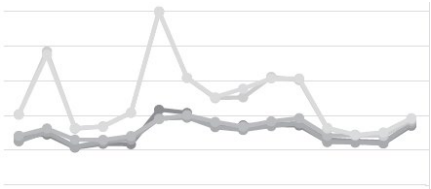
# Sammlung der Daten: Technische Umsetzung

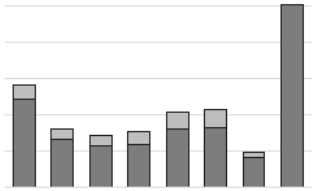
- Beschreibung der technischen Umsetzung der Messungen / verwendete Messtechniken:
  - Sind Eingriffe in Programme erforderlich, um Messdaten zu erhalten?
  - Anzahl Messwiederholungen im Fall von zufälligen/nichtdeterministischen Faktoren…
- Beschreibung der technischen Plattform, auf der die Messungen ausgeführt wurden (Betriebssystem, Prozessor/Speicher, Compiler-Version, …).

# Darstellung von Messergebnissen

- Tabellen (Auflistung sämtlicher Messwerte, Ausschnitte, …).
- Diagramme
  - Balkendiagramme
  - Tortendiagramme
  - Plots
  - Box-Plots
  - …
- Textuell (z.B. Min/Max/Durchschnittswerte).

| Coverage Criterion | Name | # Test Goals | Avg. # Test Goals per Function | CPU Time (h) | Avg. CPU Time per Function (m) | CPU Time per Goal (s) | Timeouts (%) | Avg. Speedup to VbV (Factor) | # Test Cases | Avg. # Test Cases per Function | # Covered Goals per Test Case | # Covered Goals (%) | Detected Faults (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C0 | VbV | 14,753 | 351.3 | 7.2 | 10.2 | 1.7 | 2% | 1.0 | 2417 | 57.5 | 6.1 | 98.7% | 84.1% |
| | SPL | 3,053 | 72.7 | 1.7 | 2.5 | 2.1 | 2% | 4.1 | 551 | 13.1 | 5.5 | 96.6% | 84.3% |
| | Merged1 | 6,489 | 154.5 | 3.0 | 4.2 | 1.6 | 10% | 2.4 | 1240 | 29.5 | 5.2 | 92.0% | 76.8% |
| | Merged2 | 5,052 | 120.3 | 2.8 | 4.0 | 2.0 | 12% | 2.5 | 1060 | 25.2 | 4.8 | 87.1% | 74.4% |
| | Merged3 | 4,653 | 110.8 | 3.0 | 4.3 | 2.3 | 10% | 2.4 | 976 | 23.2 | 4.8 | 87.7% | 85.7% |
| | Merged4 | 5,014 | 119.4 | 3.3 | 4.7 | 2.3 | 14% | 2.2 | 1005 | 23.9 | 5.0 | 84.7% | 70.4% |
| | Merged5 | 6,875 | 163.7 | 3.8 | 5.4 | 2.0 | 10% | 1.9 | 1066 | 25.4 | 6.4 | 80.3% | 79.2% |
| | Merged6 | 7,205 | 171.5 | 3.8 | 5.4 | 1.9 | 12% | 1.9 | 1060 | 25.2 | 6.8 | 79.3% | 81.7% |
| C3 | VbV | 10,856 | 258.5 | 7.5 | 10.7 | 2.5 | 2% | 1.0 | 2664 | 63.4 | 4.1 | 98.8% | 84.1% |
| | SPL | 1,832 | 43.6 | 1.7 | 2.5 | 3.4 | 2% | 4.3 | 521 | 12.4 | 3.5 | 96.0% | 84.3% |
| | Merged1 | 4,838 | 115.2 | 3.0 | 4.3 | 2.2 | 7% | 2.5 | 1258 | 30.0 | 3.8 | 91.1% | 76.8% |
| | Merged2 | 3,883 | 92.5 | 2.9 | 4.1 | 2.7 | 12% | 2.6 | 1075 | 25.6 | 3.6 | 83.7% | 76.8% |
| | Merged3 | 3,649 | 86.9 | 3.0 | 4.3 | 3.0 | 12% | 2.5 | 970 | 23.1 | 3.8 | 81.6% | 83.7% |
| | Merged4 | 3,406 | 81.1 | 3.3 | 4.7 | 3.5 | 14% | 2.3 | 944 | 22.5 | 3.6 | 78.5% | 79.6% |
| | Merged5 | 4,462 | 106.2 | 3.7 | 5.3 | 3.0 | 14% | 2.0 | 987 | 23.5 | 4.5 | 73.5% | 86.8% |
| | Merged6 | 4,644 | 110.6 | 3.8 | 5.4 | 2.9 | 10% | 2.0 | 999 | 23.8 | 4.6 | 72.0% | 80.7% |





# Reproduzierbarkeit von Messergebnissen

Idealfall: Es wird eine Webseite bereitgestellt mit

- einer virtuellen Maschine, in der automatisiert die Experimente unter „Originalbedingungen“ auf beliebigen Plattformen wiederholt werden können (mitsamt Installationsanleitung).
- dem Quellcode der Implementierung von „M“.
- dem Quellcode und die vollständige Dokumentation des verwendeten Korpus.

# Diskussion von Messergebnissen

- „Wertfreie“ Beschreibung, was in der Darstellung der Messergebnisse zu sehen ist.
- Beschreibung allgemeiner Tendenzen („Die Reduktion der Testsuite beträgt durchschnittlich 25%.“)
- Ermittlung etwaiger mathematischer Zusammenhänge zwischen Größen („Testdauer fällt linear mit der Testsuite-Größe…“)
- Herausgreifen von interessanten Min/Max-Werten.
- Ergebnisse statistischer Signifikanztests.
- Erklärung von Ausreißern und Ausnahmen.
- Noch **KEIN** Bezug auf die Hypothesen/Forschungsfragen

# Beispiel: Diskussion von Messergebnissen

- Bestätigung oder Widerlegung der Hypothesen bzw. Beantwortung der Forschungsfragen auf Basis der Messergebnisse.
- Dabei muss die gewählte Darstellung und Beschreibung der Messergebnisse diese Schlussfolgerungen eindeutig und nachvollziehbar unterstützen und aufzeigen.

# Validität von Schlussfolgerungen

- Den Abschluss bildet ein Diskussion möglicher **Gefährdungen der Belastbarkeit und Verallgemeinerbarkeit** der Ergebnisse.
- Beispiele:
  - Ist die Korpus-Auswahl hinreichend repräsentativ, um schlussfolgern zu können, dass ähnliche Ergebnisse auch für andere Programme zu erwarten sind?
  - Ist die Anzahl redundanter Testfälle in den ursprünglichen Testsuiten realistisch?
  - Sind Mutanten ein geeignetes Mittel zur Bewertung der Effektivität?
  - Gelten die Ergebnisse auch für Integrationstests?
  - …

# Literatur


- Jutta Markgraf, Hans-Peter Musahl, Friedrich Wilkening, Karin Wilkening, and Viktor Sarris. *Studieneinheit Versuchsplanung*, 2001. FIM-Psychologie Modellversuch, Universität Erlangen-Nürnberg.
- Jürgen Bortz. *Statistik für Human- und Sozialwissenschaftler*. Springer, 2004. http://www.springer.com/psychology/book/978-3-642-12769-4?changeHeader
- Robert A. Donnelly Jr. *The Complete Idiot's Guide to Statistics*. Alpha, 2007