Test-Suite Beurteilung

Seminar: Beiträge zum Software Engineering
@FU Berlin, Prof. Dr. Lutz Prechelt

WS14/15 - Mehmet Bayram

Verglichene Arbeiten

A

Comparing Non-adequate Test Suites using Coverage Criteria

Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, Darko Marinov

Univ. of Illinois at Urbana-Champaign
Oregon State University

2013 International Symposium on Software Testing and Analysis

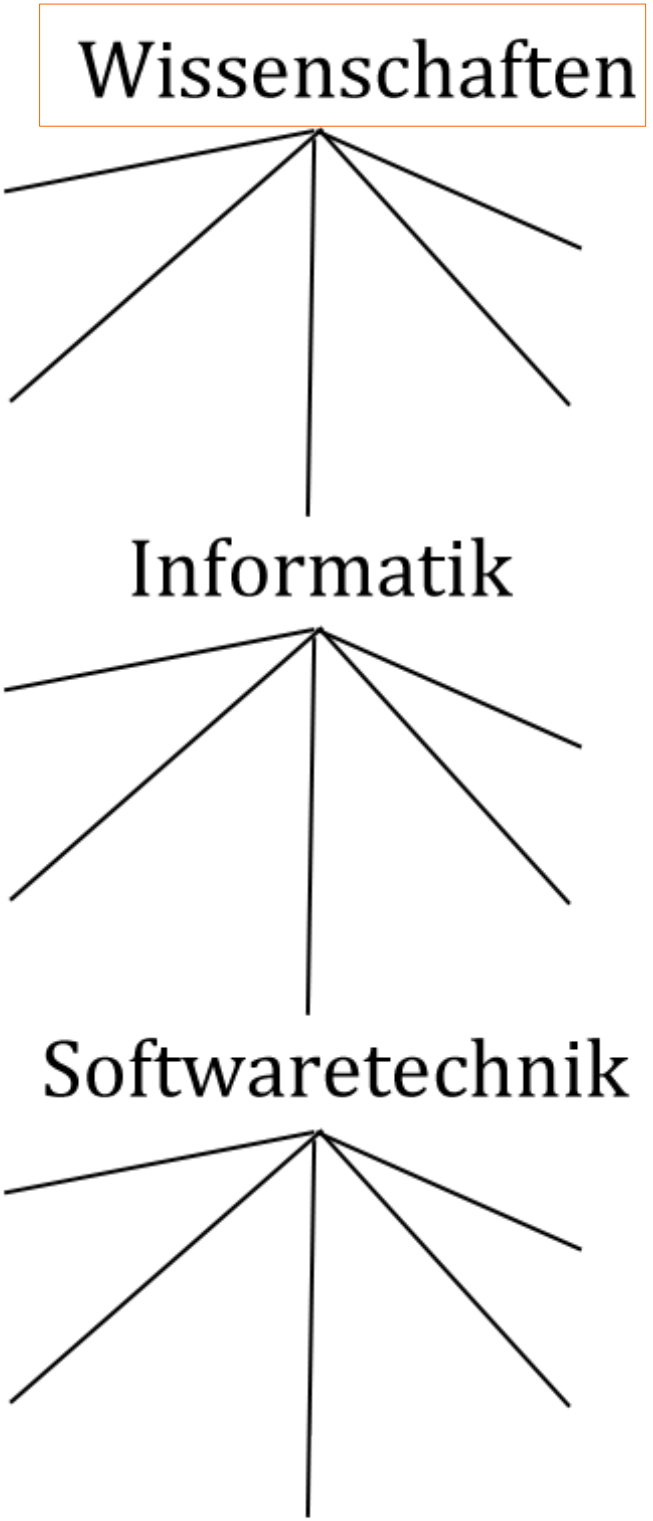
B

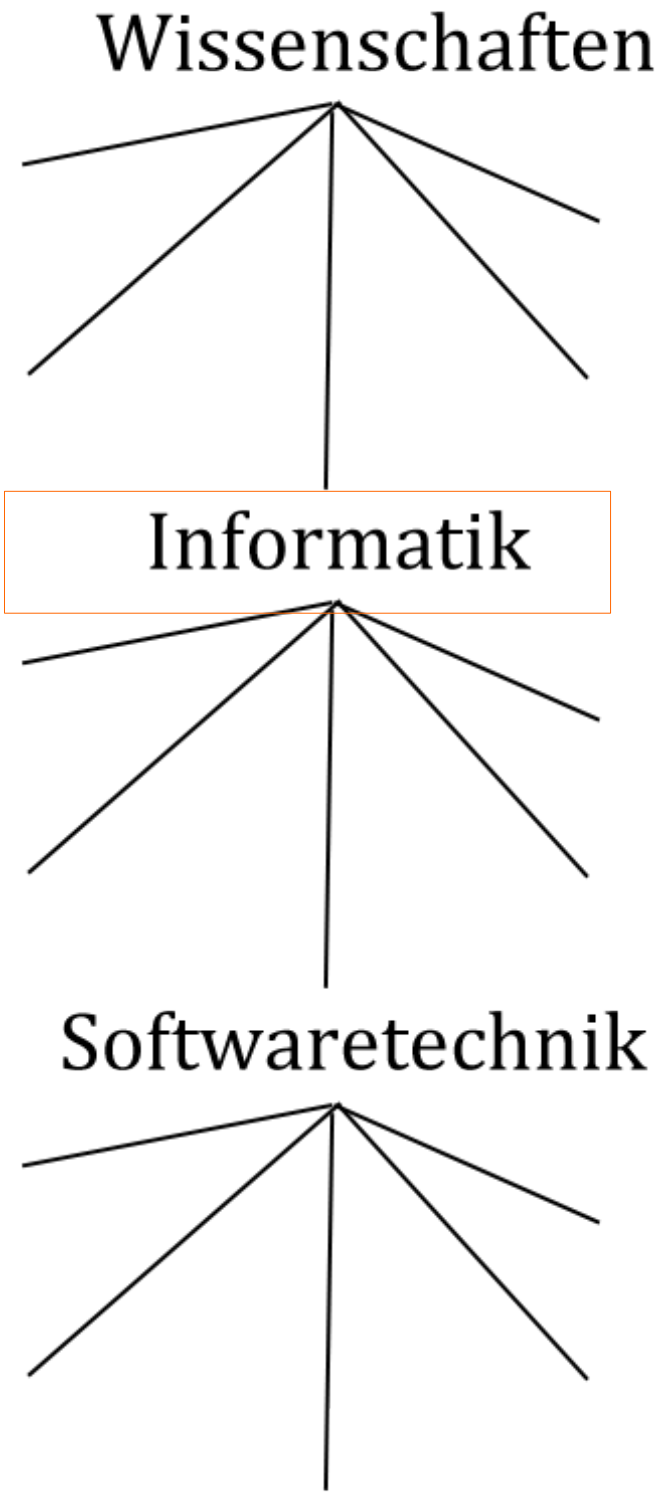
Code Coverage for Suite Evaluation by Developers

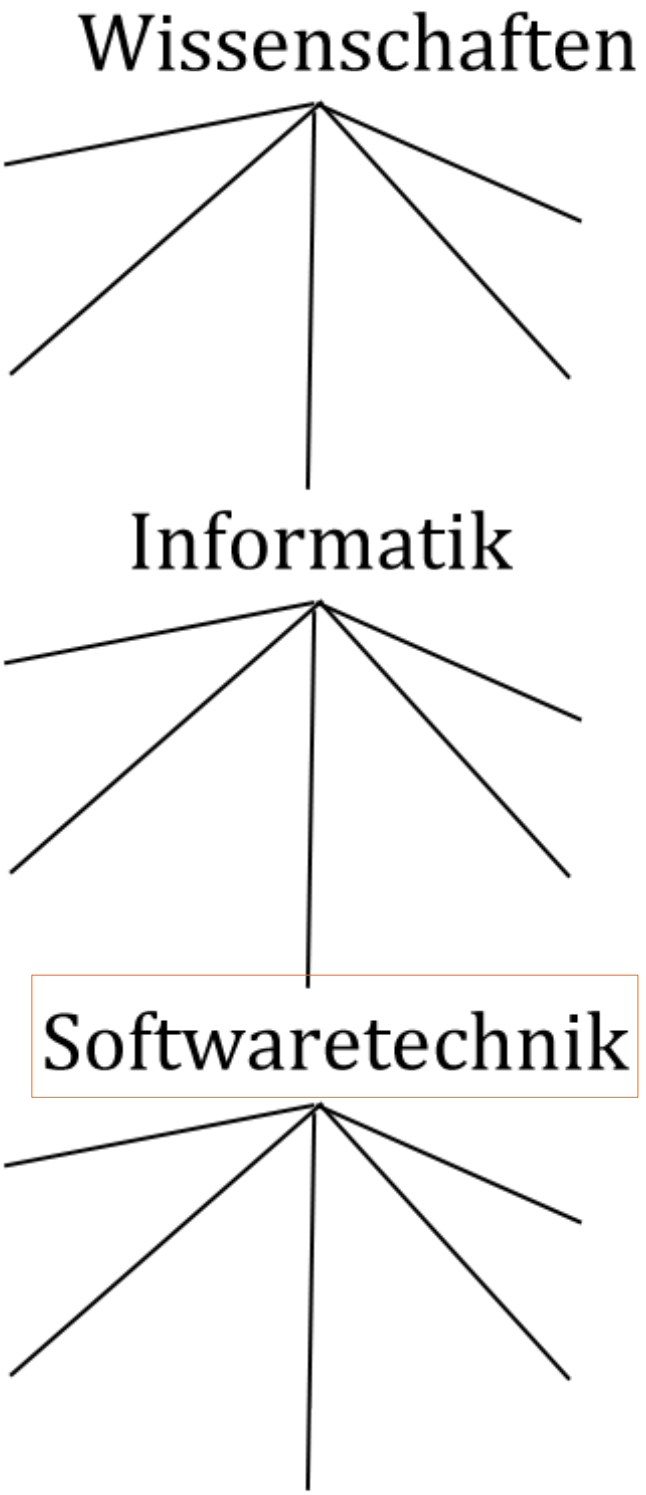
Rahul Gopinath
Carlos Jensen
Alex Groce

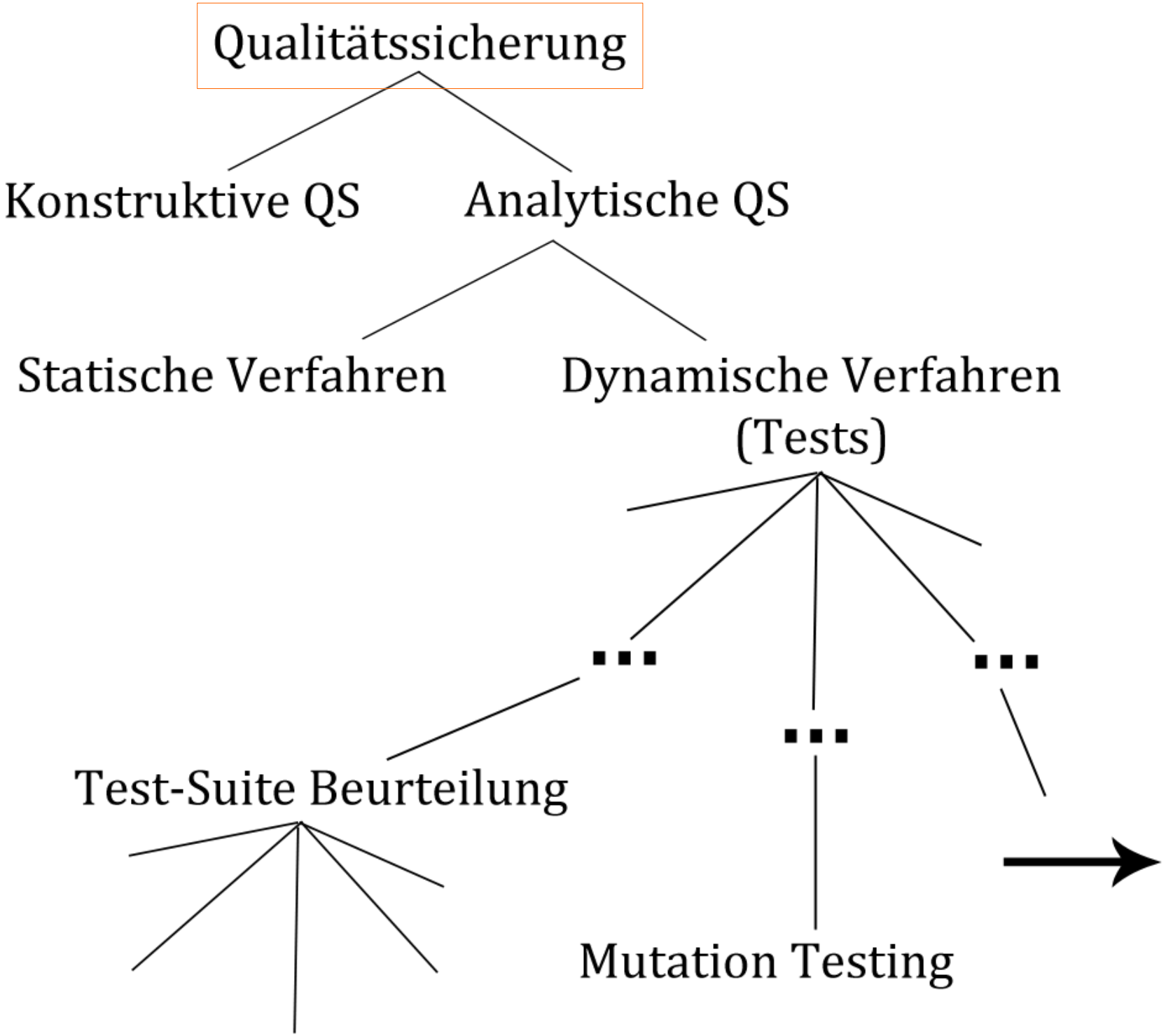
Oregon State Univ. / USA

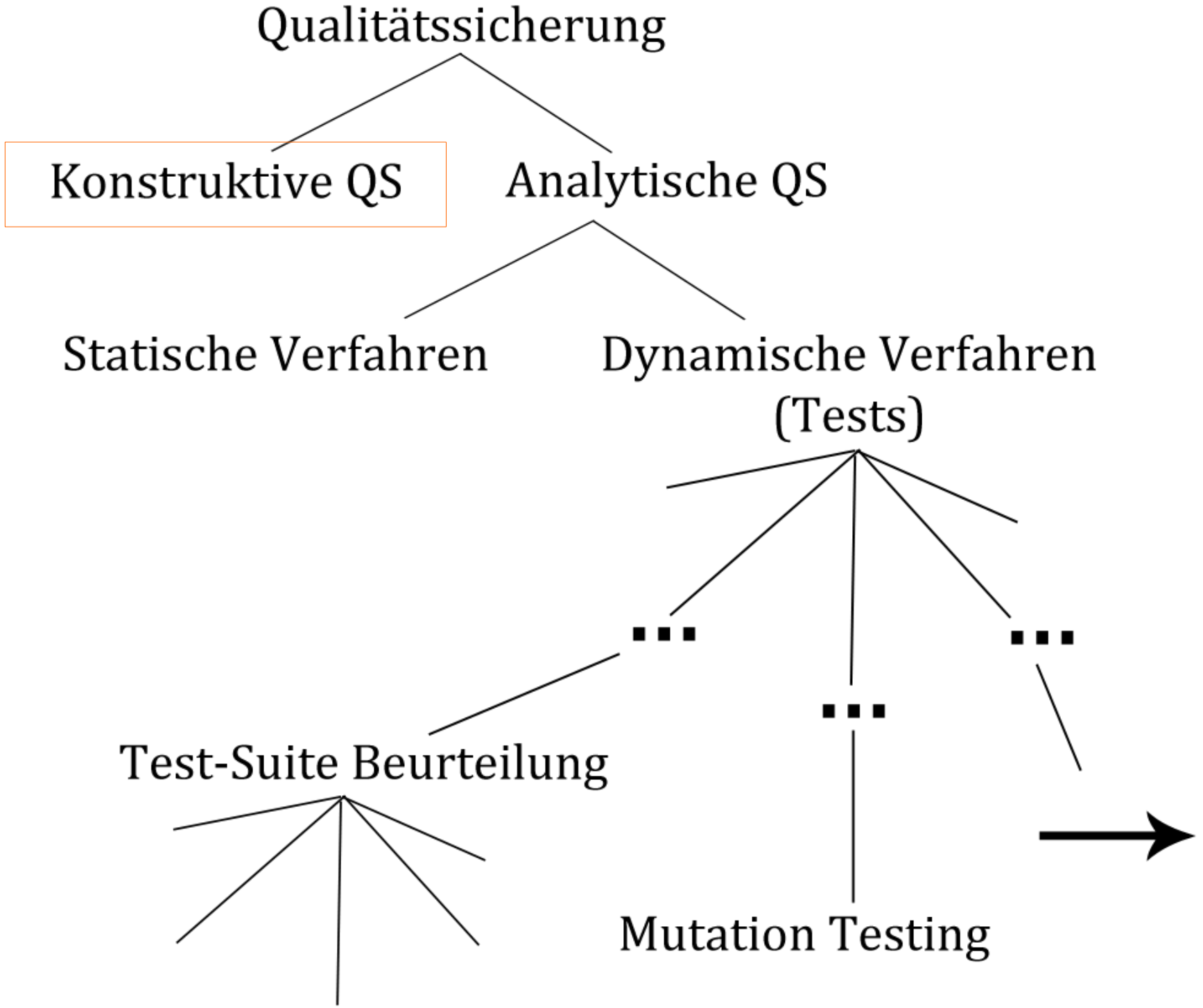
2014 International Conference on Software Engineering (ICSE)

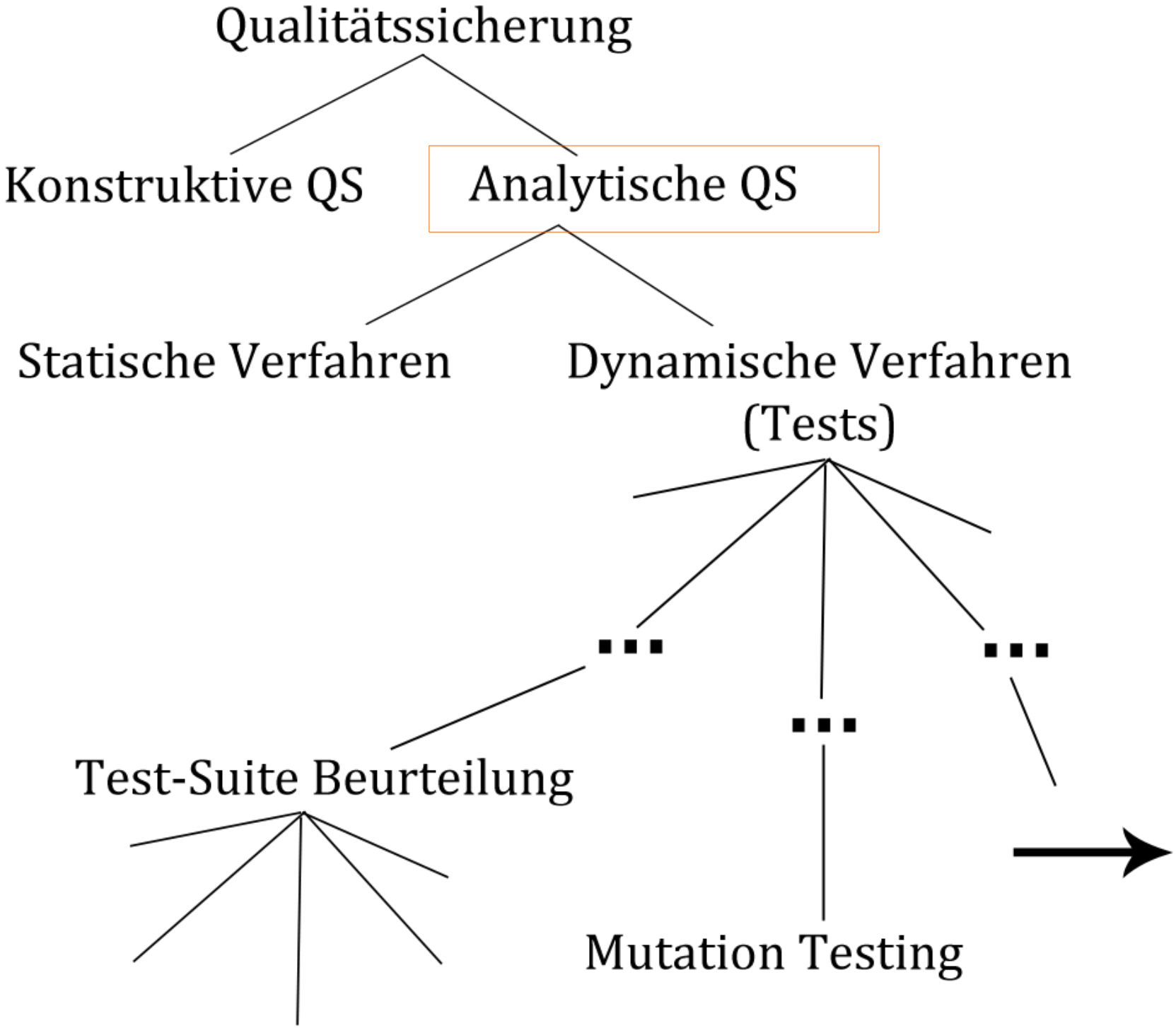

Wissenschaften
Informatik
Softwaretechnik

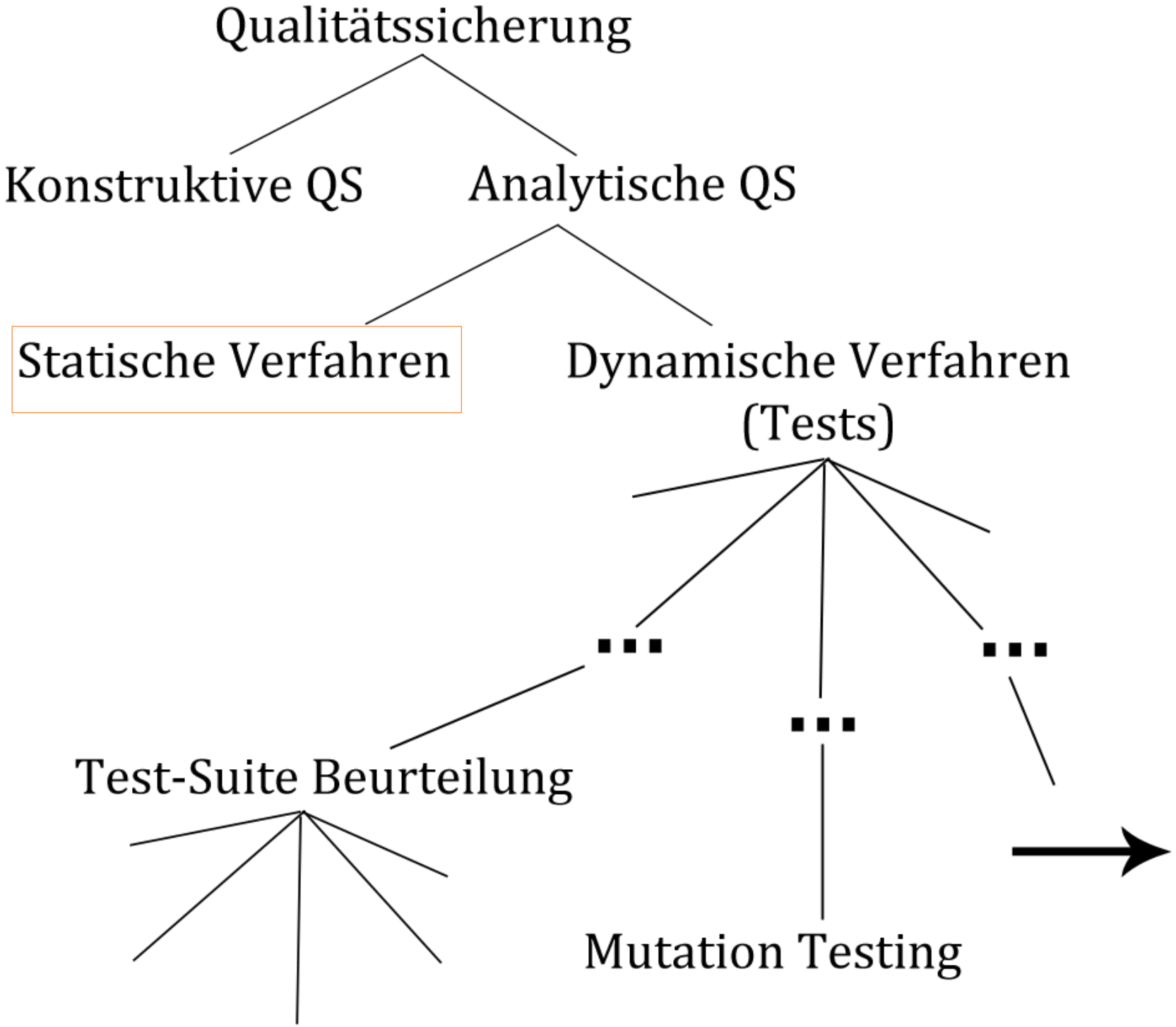

Wissenschaften
Informatik
Softwaretechnik

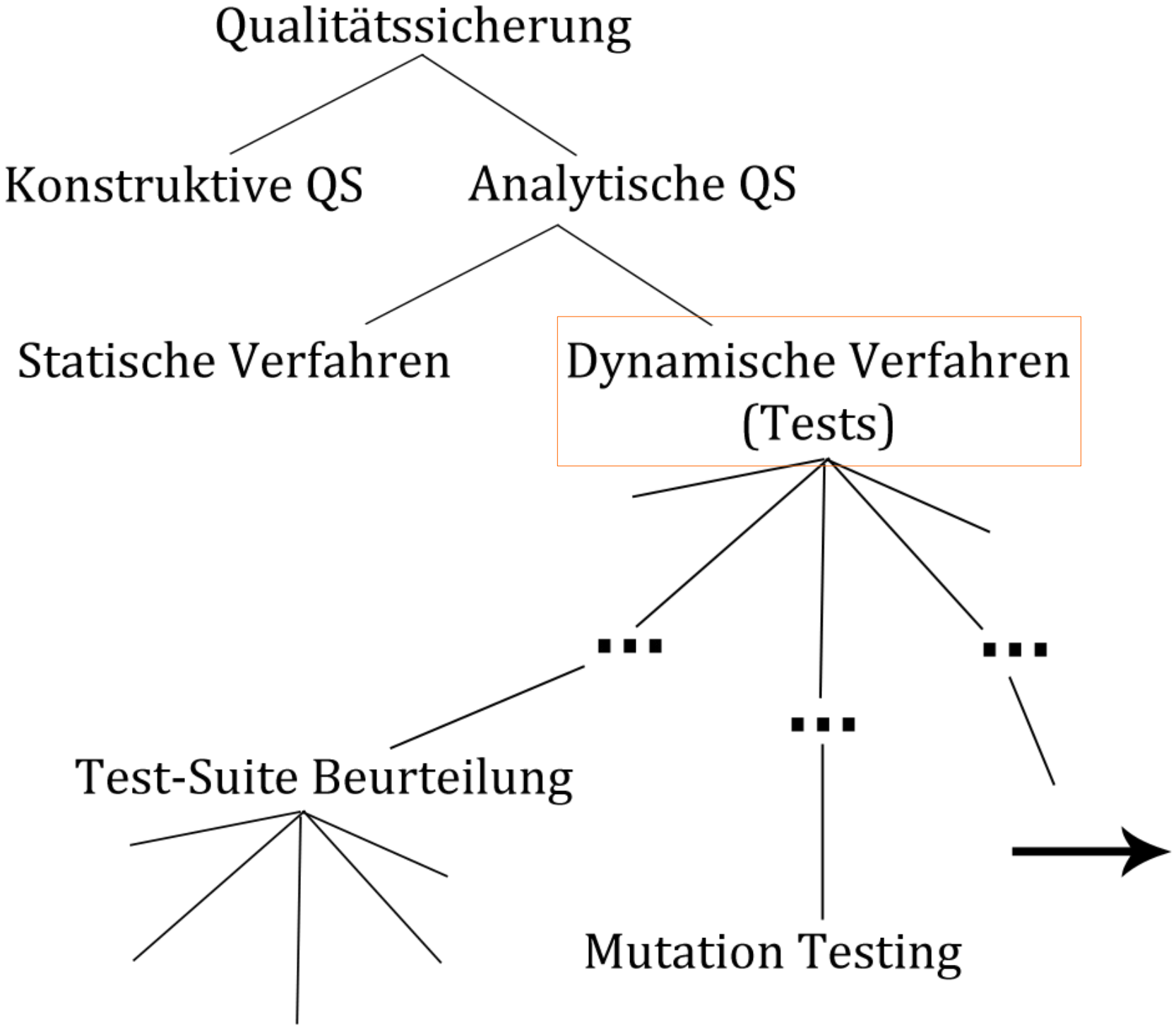

Wissenschaften
Informatik
Softwaretechnik

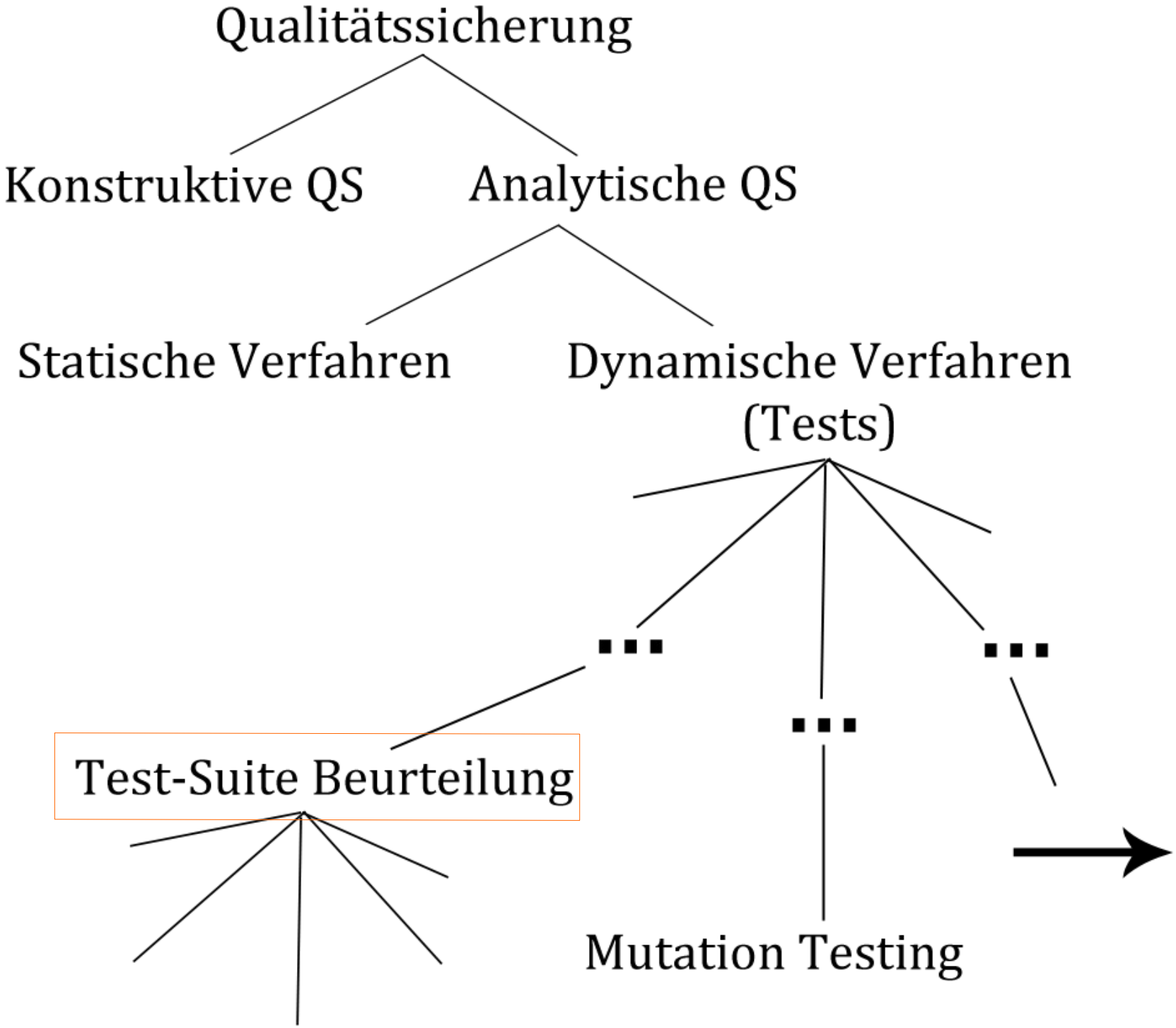

Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing


Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing


Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing

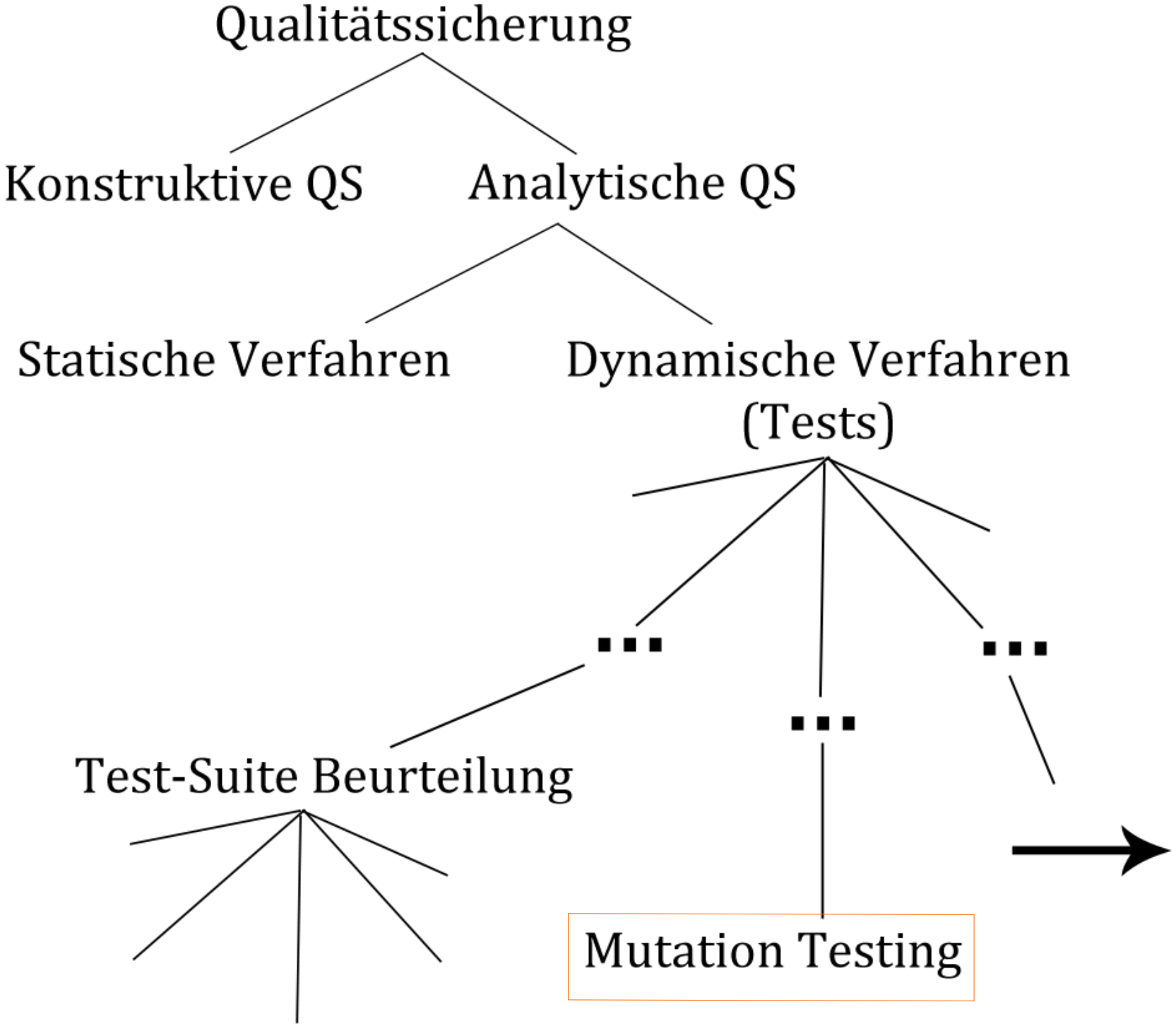

Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing


Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing


Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing

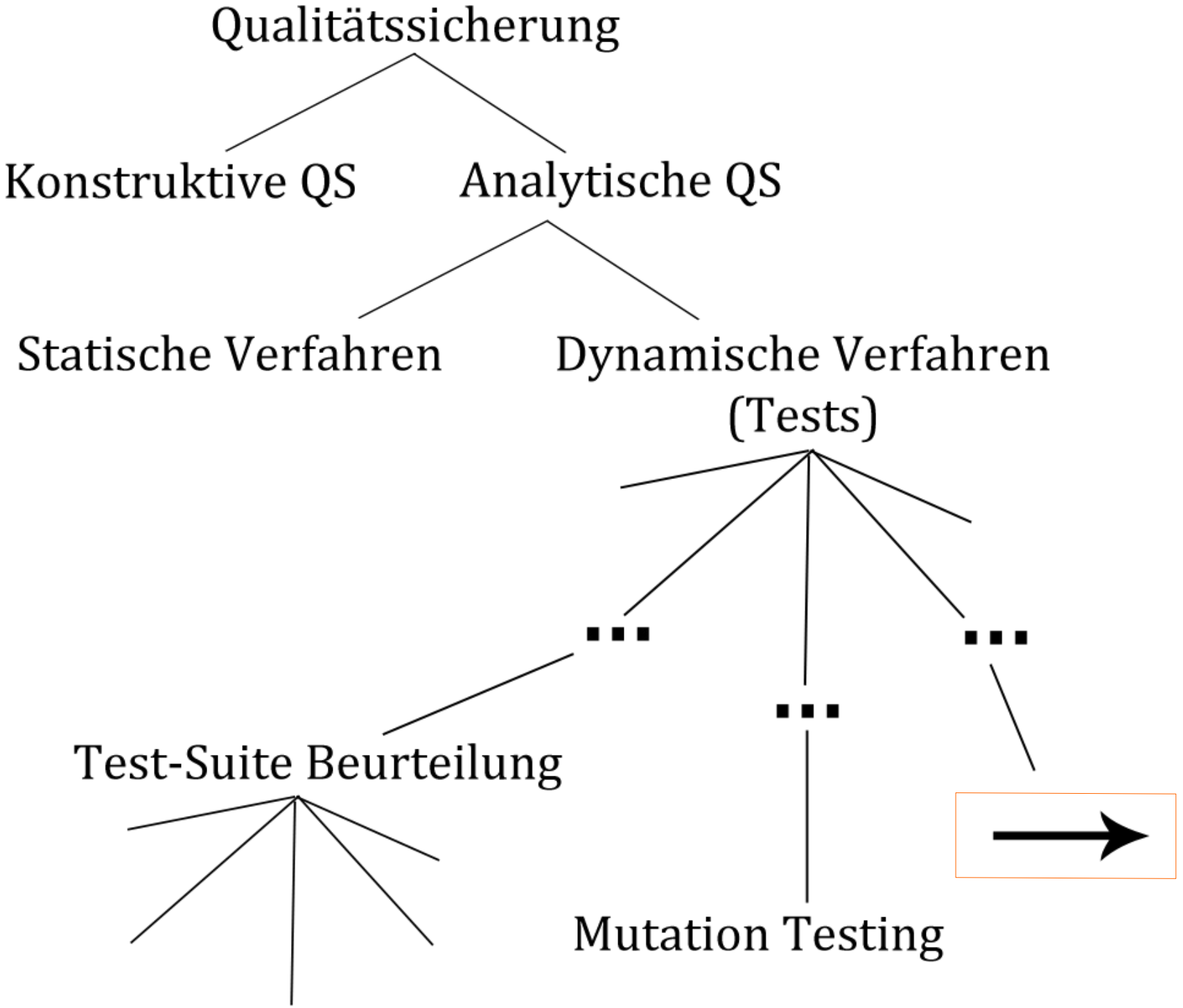
Test-Suite Beurteilung

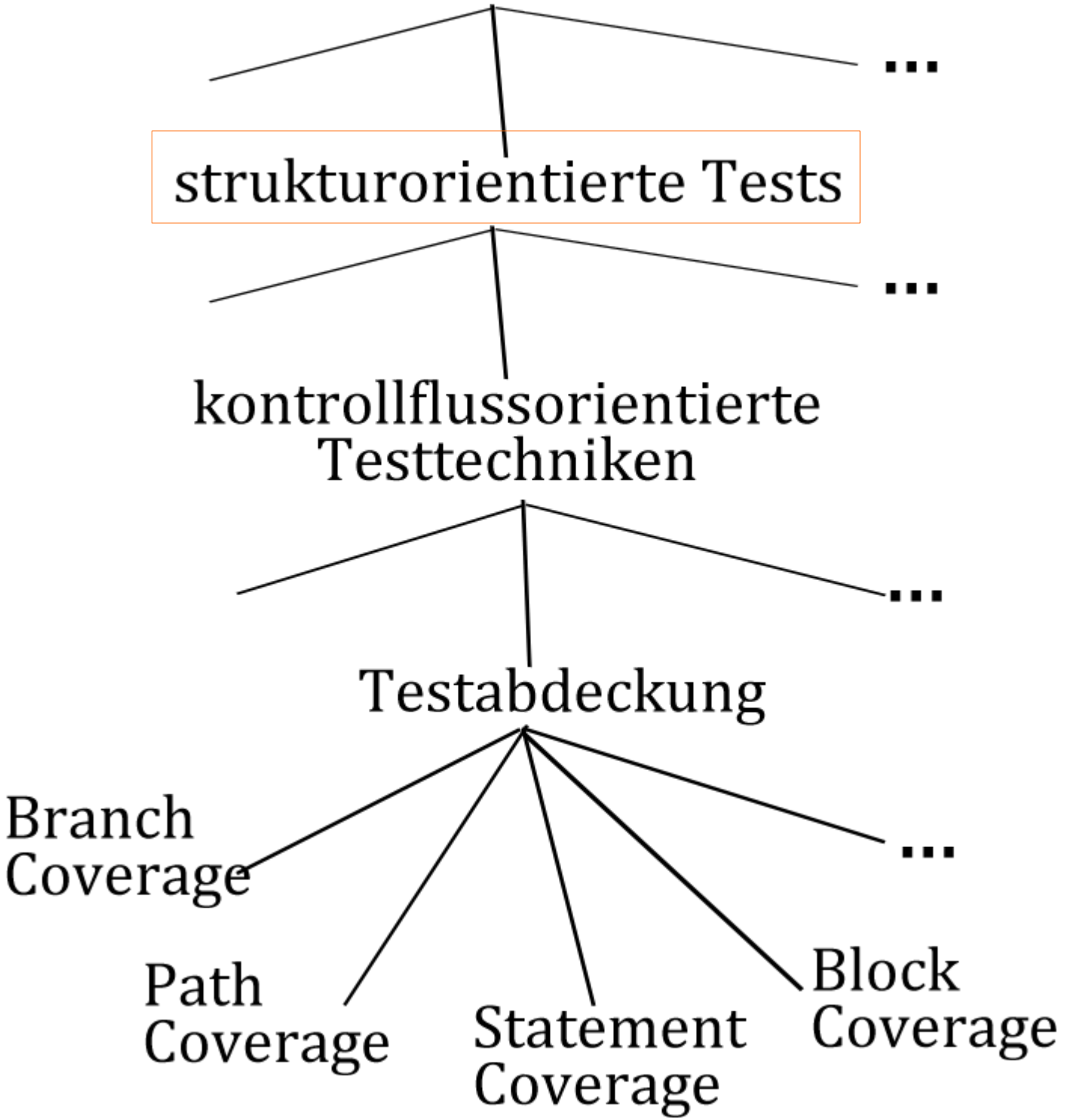
- Test-Suite: Menge von Testfällen
- Testfall: elementares Test, überprüft bestimmte Eigenschaften
- Ziele:
  - Möglichst gute Test-Suites erstellen
  - Test-Suites gut auswählen
  - Test-Suite Generatoren testen
  - Techniken zur Generierung Test-Suite testen
- Maßnahme: Vorhandene Test-Suites beurteilen
- Maß: Fähigkeit, Fehler zu finden
- Problem: Wie messen, wie viele der Fehler gefunden wurden?
- Andere Lösungswege zur Beurteilung notwendig...

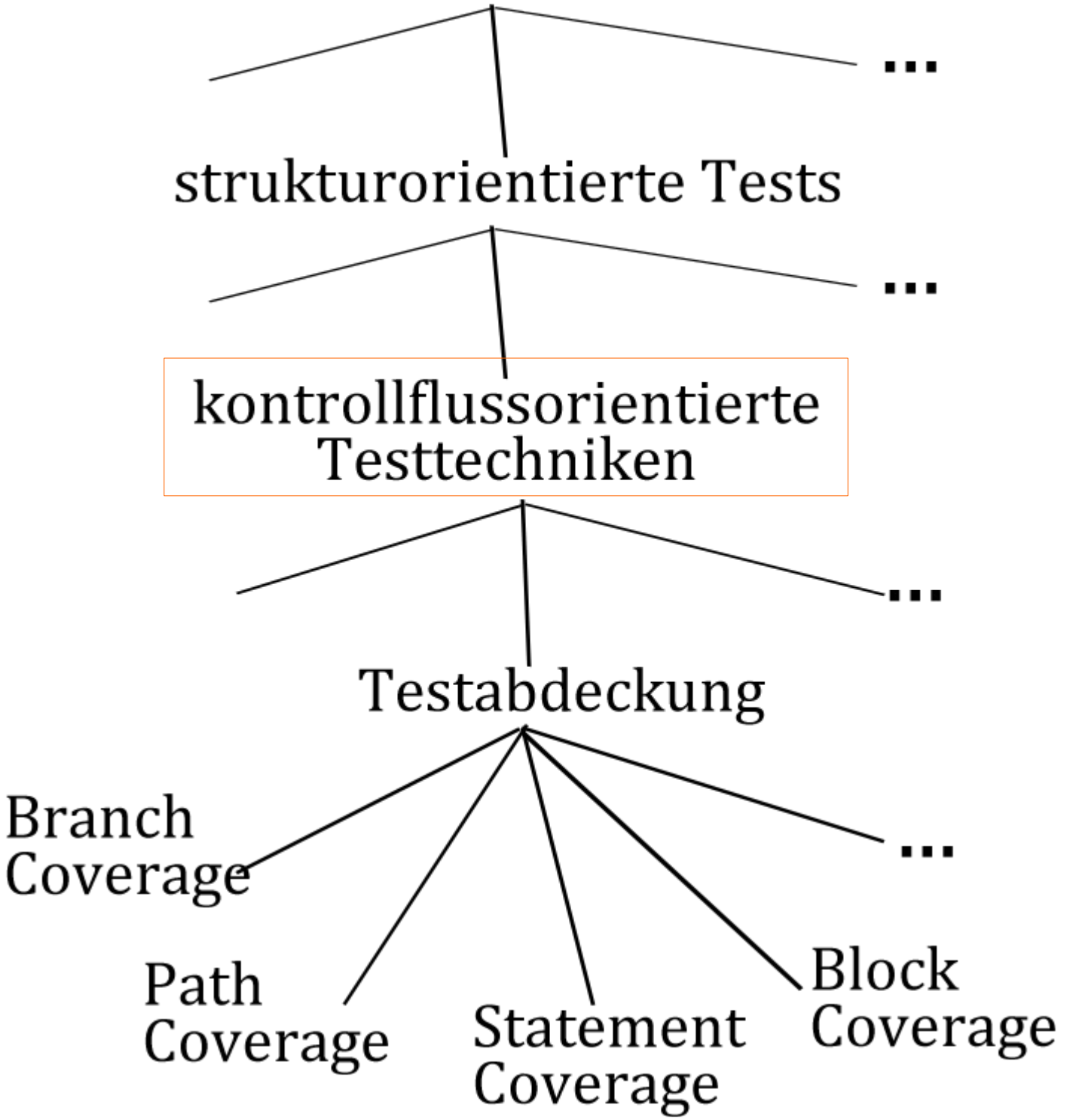

Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing

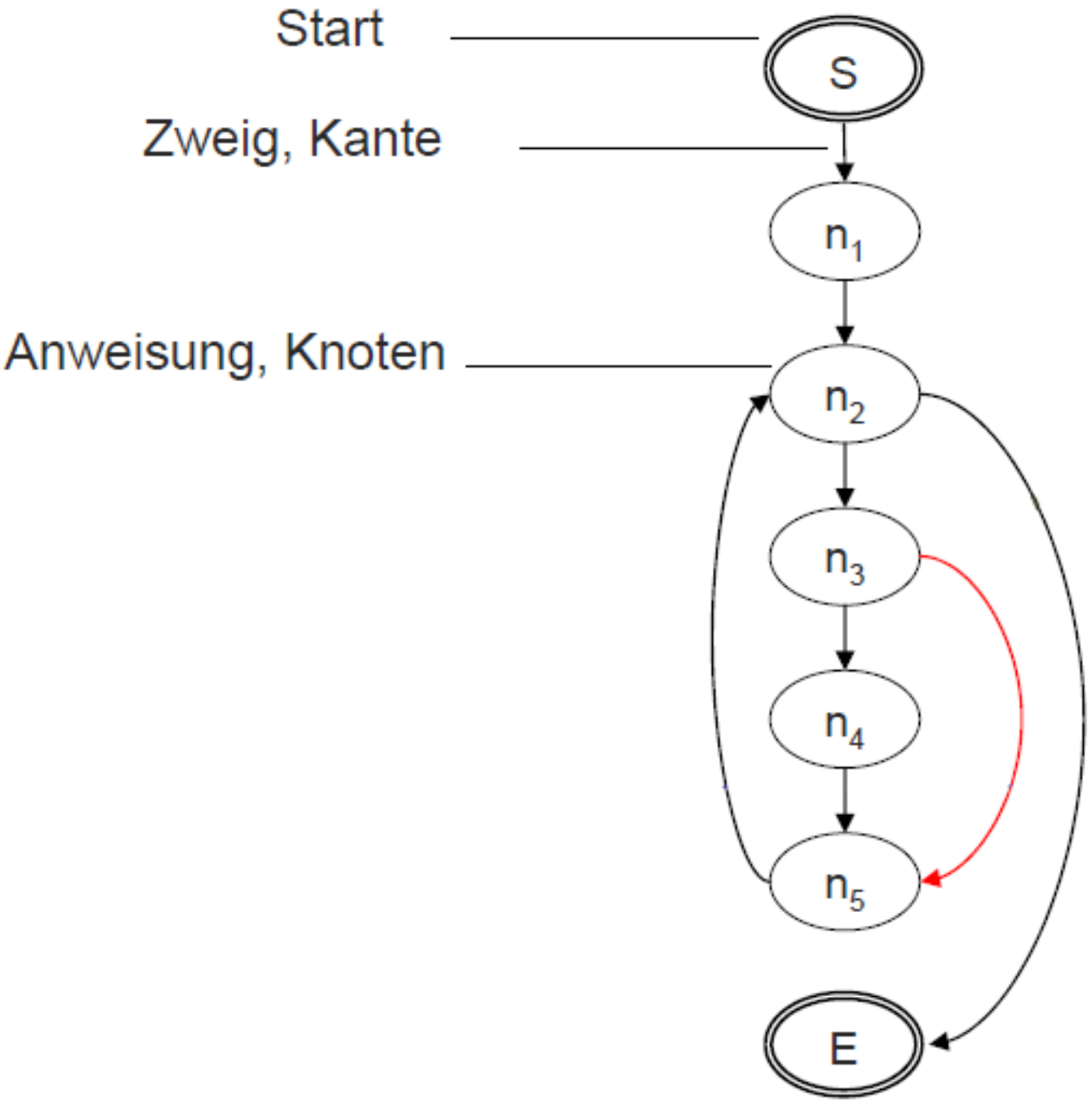
Mutation Testing

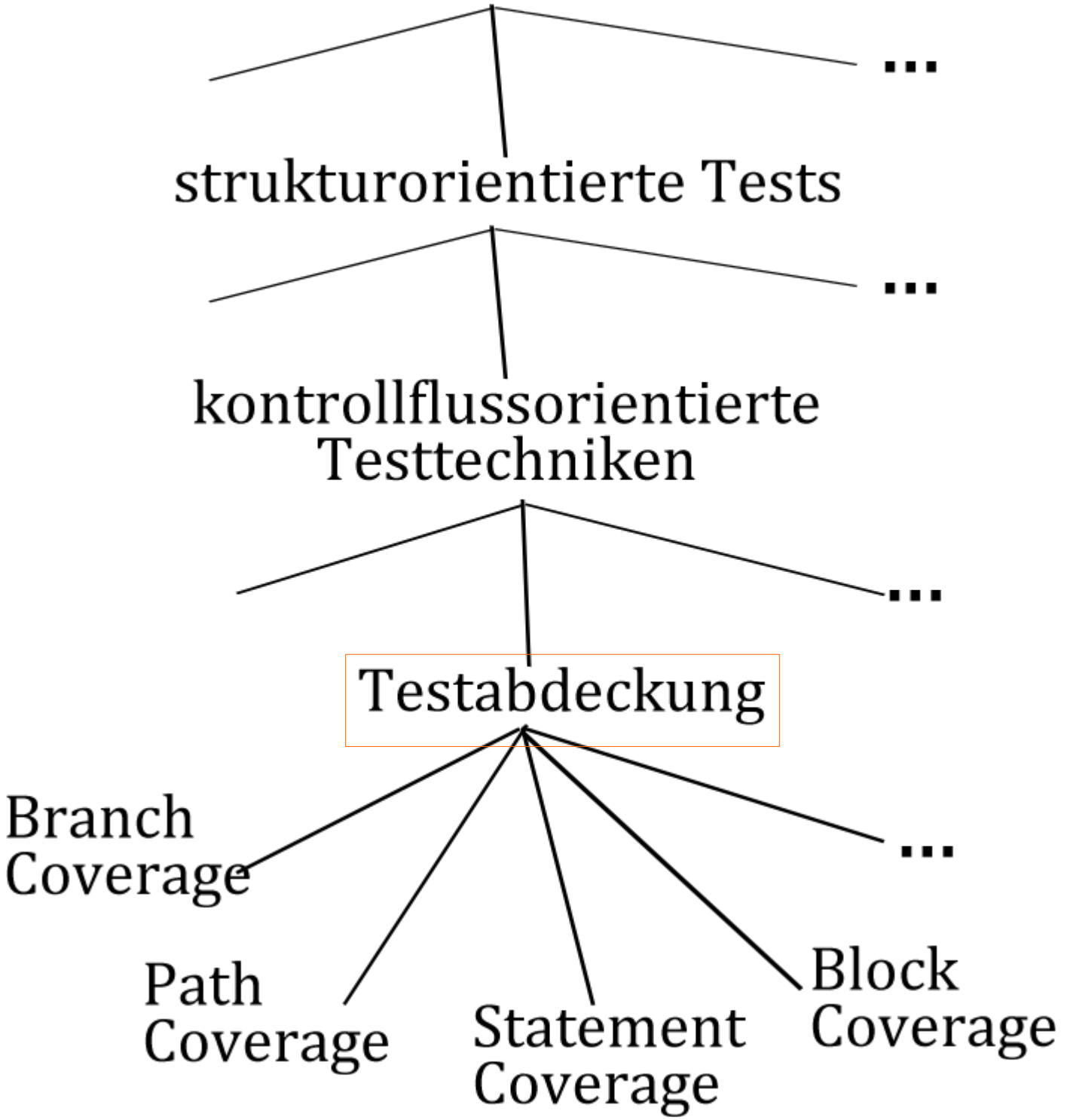
- Vorgehensweise:
  - Manipuliere Programm an vielen Stellen: z.B. == → !=
  - eingebaute Fehler wie natürliche Fehler
  - Test-Suite, die diese Fehler findet, hätte damit auch natürliche Fehler gefunden → Test-Suite Beurteilung
- Weiterer Anwendungsfall: unwirksamen Code entdecken
- Probleme
  - komplizierter
  - sehr viele Ausführungskombinationen → Laufzeit
- Fazit: Sehr gut, aber schwierig umsetzbar
- **B:** Mehrere Arbeiten: Mutual-Testing besser als Abdeckung keine Arbeit zeigte bisher das Gegenteil


Qualitätssicherung
Konstruktive QS
Analytische QS
Statische Verfahren
Dynamische Verfahren
(Tests)
Test-Suite Beurteilung
Mutation Testing


...
strukturorientierte Tests
...
kontrollflussorientierte Testtechniken
...
Testabdeckung
Branch Coverage
...
Path Coverage
Statement Coverage
Block Coverage


...
strukturorientierte Tests
...
kontrollflussorientierte Testtechniken
...
Testabdeckung
Branch Coverage
...
Path Coverage
Statement Coverage
Block Coverage


Start
Zweig, Kante
Anweisung, Knoten
S
n_1
n_2
n_3
n_4
n_5
E


...
strukturorientierte Tests
...
kontrollflussorientierte
Testtechniken
...
Testabdeckung
Branch
Coverage
...
Path
Coverage
Statement
Coverage
Block
Coverage

Abdeckungskriterien

- Idee: möglichst alle _____ des Programms ansprechen
- Vorteil: Software während der Entwicklung schon testen
- Vollständigkeit des Testens messbar
- Findet unwirksamen Code
- Test-Suite bei Kriterium K 100% Abdeckung → K-adequat

Forschungsfrage

- Gemeinsame Aussagen:
  - Test-Suite Beurteilung sehr wichtige Aufgabe
  - Mutation Testing leistet sehr gute Arbeit
  - Mutation Testing schwierig anwendbar → Alternativen
  - Abdeckungskriterien bieten sich sehr gut an
- Gemeinsame Forschungsfrage:
  Welches Abdeckungskriterium leistet Mutation-Test Ergebnisse und ist dennoch leicht anwendbar?
- Vorgehensweise: experimentelles Vergleichen

Einbezogene Abdeckungskriterien

A

- **Branch Coverage (BC)**
- **Statement Coverage (SC)**
- **Intra-Method Path Coverage (IMP)**
- **Acyclic Intra-Method Path Coverage (AIMP)**
- **Predicate Complete Test Coverage (PCT)**

B

- **Branch Coverage (BC)**
- **Path Coverage (PC)**
- **Statement Coverage (SC)**
- **Block Coverage**

Statement Coverage / Anweisungsüberdeckung

- Ziel: Jede Anweisung mindestens 1x ausführen
- Einfach anwendbar
- Beispiel: if (condition) {statement1;} else {statement2;}
- Problem: Besucht alle Knoten,
  aber durchläuft nicht alle Kanten
- Zu schwach, um als Hauptinstrument einzusetzen

Branch Coverage / Zweigüberdeckungstest

- Enthält Anweisungsüberdeckung
- Wird als Minimalkriterium betrachtet
- Idee: Alle Zweige des Programms ausführen
  → Alle Kanten werden durchlaufen
- Findet nicht ausführbare Programmzweige
  Keine Testdaten generierbar, die Zweige ausführen lassen
- Probleme:
  - Kombinationen von Verzweigungen nicht berücksichtigt
  - Schleifen werden nicht ausreichend getestet

Path Coverage / Pfadabdeckung

- Ziel: Alle möglichen Pfade durchführen
- Beinhaltet Anweisungs- & Verzweigungsüberdeckung
- Schleifen: Jeder Durchlauf neuer Pfad
- Sehr umfassend → oft nicht vollständig realisierbar
- falls realisiert → sehr hohe Erfolgsrate

Block Coverage / Blockabdeckung

- Sehr ähnlich zu Anweisungsüberdeckung
- Betrachtet nicht einzelne Anweisungen, sondern Blöcke
- Bsp: if-else, Schleifen, switch-case
- Leicht schwächer als Anweisungsüberdeckung
  Beispiel:

```
if (condition1){
    statement1;
    if (condition2){
        break;
    }
    statement;
}
```

Intra Method Path Coverage

- Pfadabdeckung innerhalb von Methoden
- Beinhaltet Zweigs- & Anweisungsabdeckung
- Der Pfad wird gebildet

  vom Beginn der jeweiligen Methode

  bis zum Rücksprung von der Methode
- Andere Methodenaufrufe werden nicht einbezogen

Acyclic Intra Method Path Coverage

- Modifizierung von Intra Method Path Coverage
- Betrachtet nur Pfade, die keine Zyklen enthalten
- Reduziert Anzahl von Pfaden auf maximal M 2^K
  M Anzahl Methoden, K max. Anzahl Verzweigungen

Predicate Complete Test Coverage PCT

- Programm aus
  M Anweisungen und
  N elementaren Prädikaten wie (a>0)
  nicht wie (b==1 && c==0)
- Ziel: Alle möglichen Zustände erreichen
- Idee: selbe Anweisung, anderes Verhalten in Ci und Cj
- Jede Anweisung in jedem Zustand testen
- Beinhaltet Anweisungsüberdeckung, Zweigsüberdeckung

Methodik

A

- 26 Programme: 15x Java, 11x C
  - 13x Java: Implementierungen von Datenstrukturen, die für Test-Forschung eingesetzt werden
  - andere Programme: verschiedene Typen
- Test Suites für Java: aus Test-Suite Pools
- Test Suites für C: anderes Pool
- für paar Programme selbst erstellt
- verschiedene Werkzeuge zur Mutanten-Erstellung
- je nach Programm unterschiedliche Test-Suite Anzahl
- Nicht alle Kriterien bei allen Programmen gemessen

Methodik

B

- Programme aus Open Source Repositiories
- große Vielfalt, alle möglichen Programme
- Programme aus der realen Welt
- Programme enthielten auch eigene Test-Suites
- Automatisiert und per Hand generierte Test-Suites
- Ziele:
  - reale Test-Suites (nicht von Forschern)
  - sign. Ergebnis → konstante Variablen → nur Java

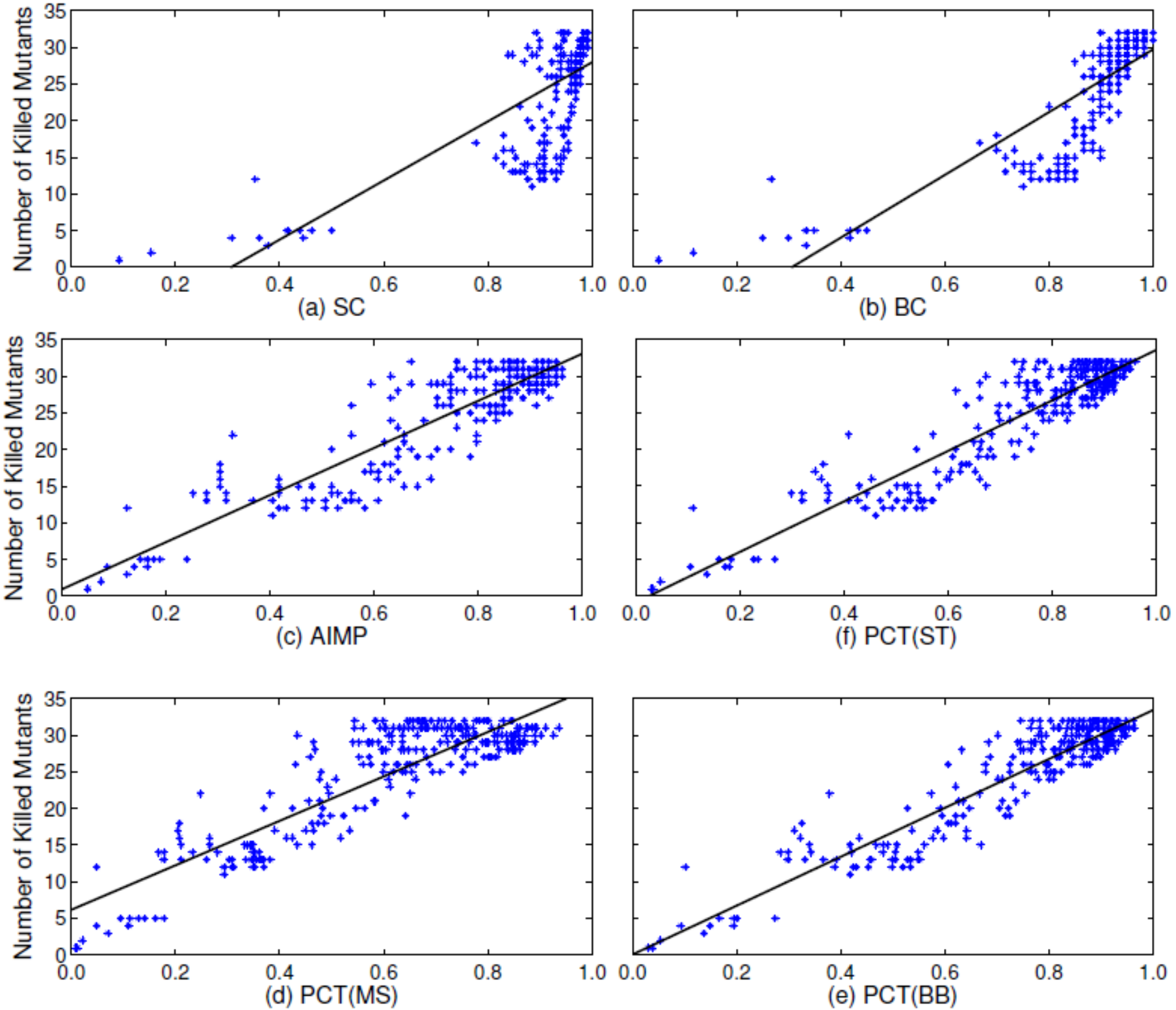
Methodik

B

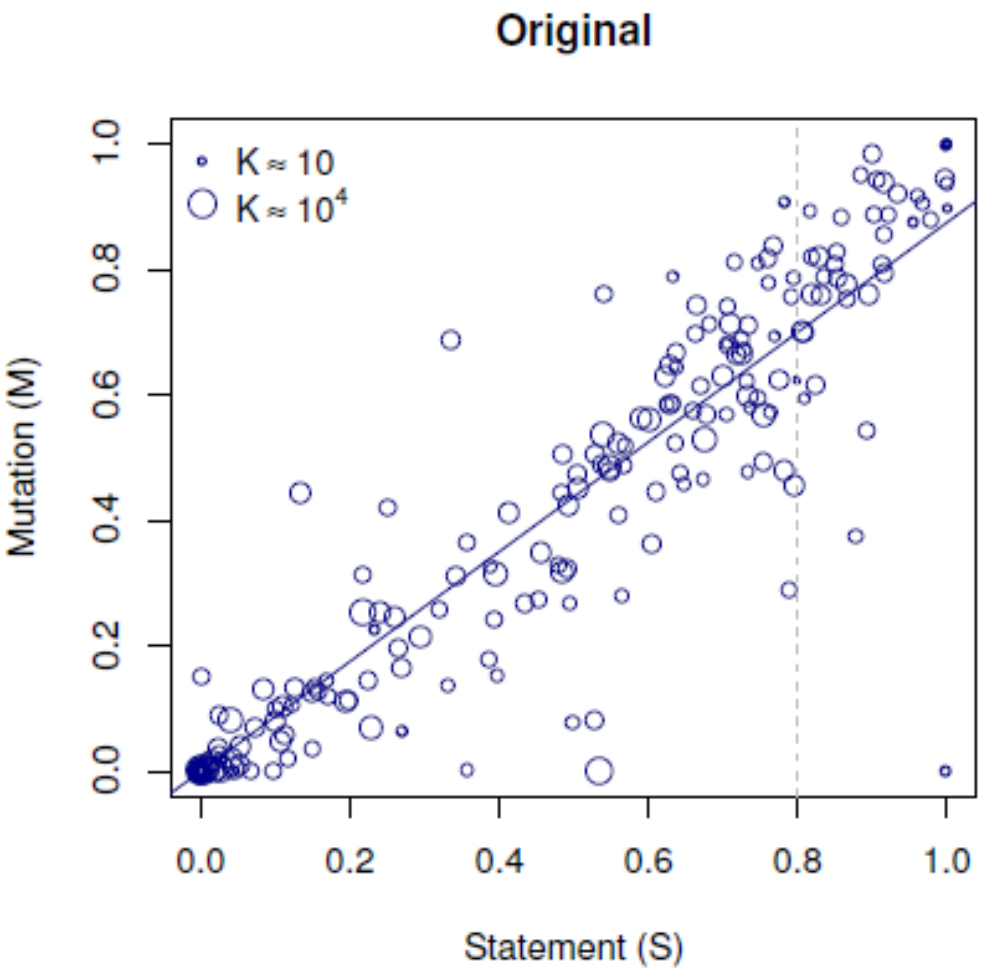
- Mehrheit von Programmiersprachen ähnlich zu Java
- Um Verzerrungen zu erkennen, zweite Phase

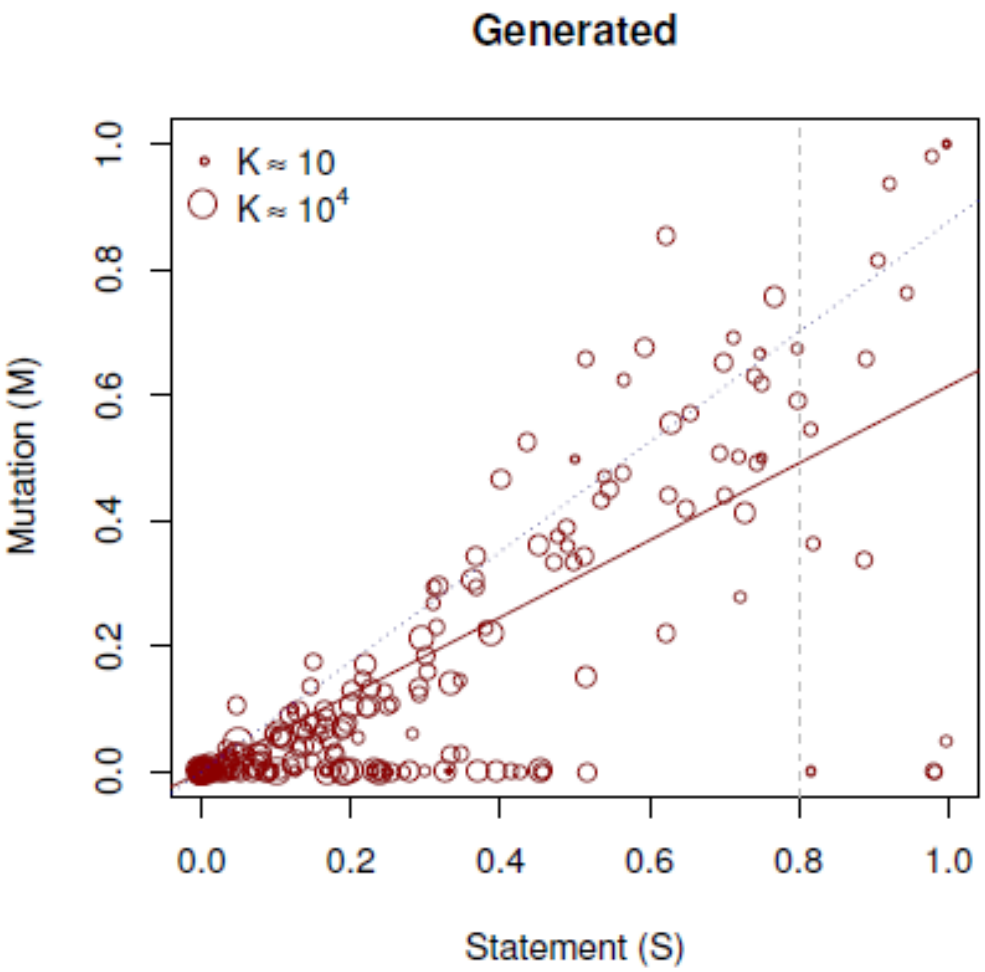
  mit selbst automatisiert generierten Test-Suites
- 729 Projekte mit eigenen Test-Suites
- ca 500 davon nicht wie gewollt testbar (z.B. Compile Fehler)
- Etwa 200 in beiden Phasen getestet

A

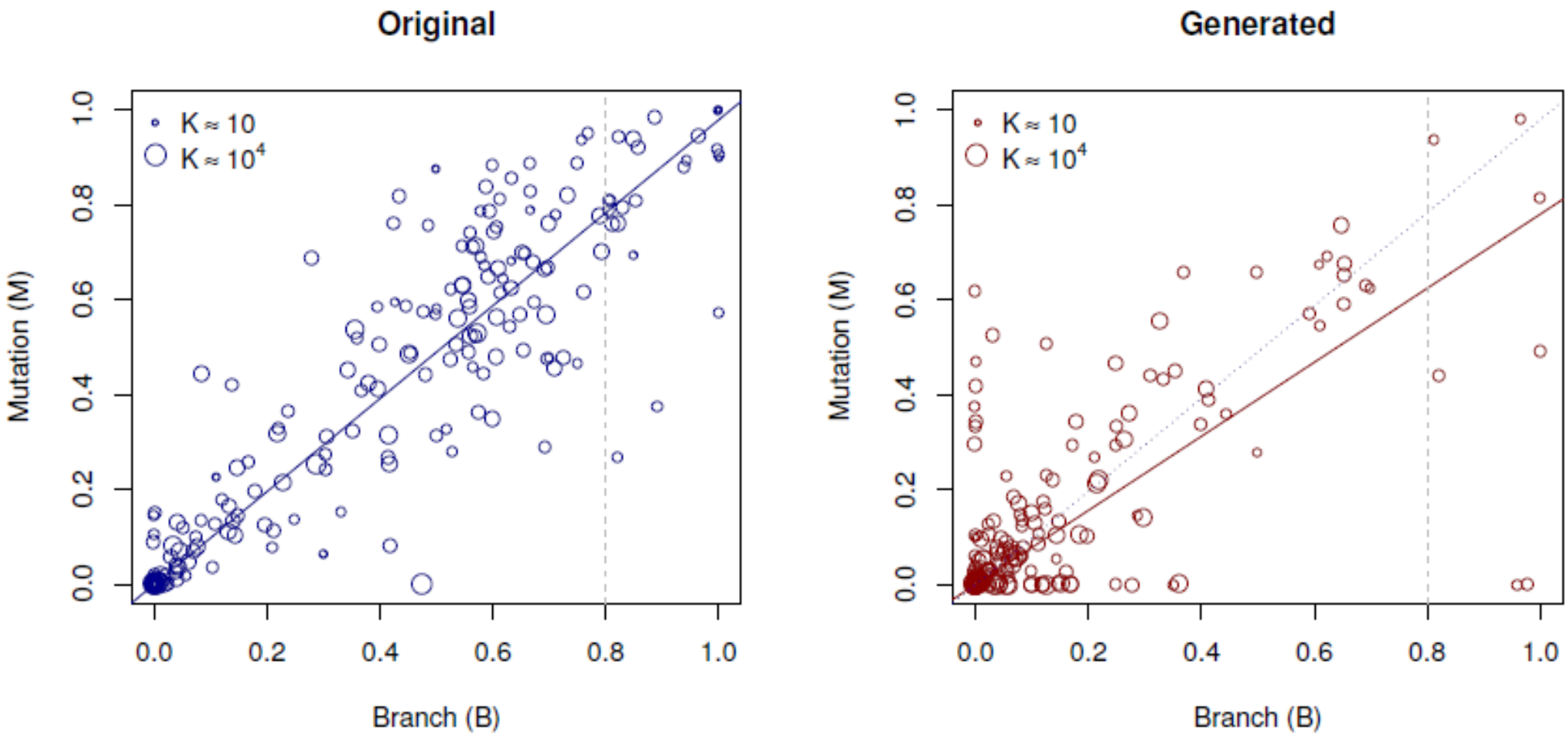

Number of Killed Mutants
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0
(a) SC
(b) BC
(c) AIMP
(f) PCT(ST)
(d) PCT(MS)
(e) PCT(BB)

B

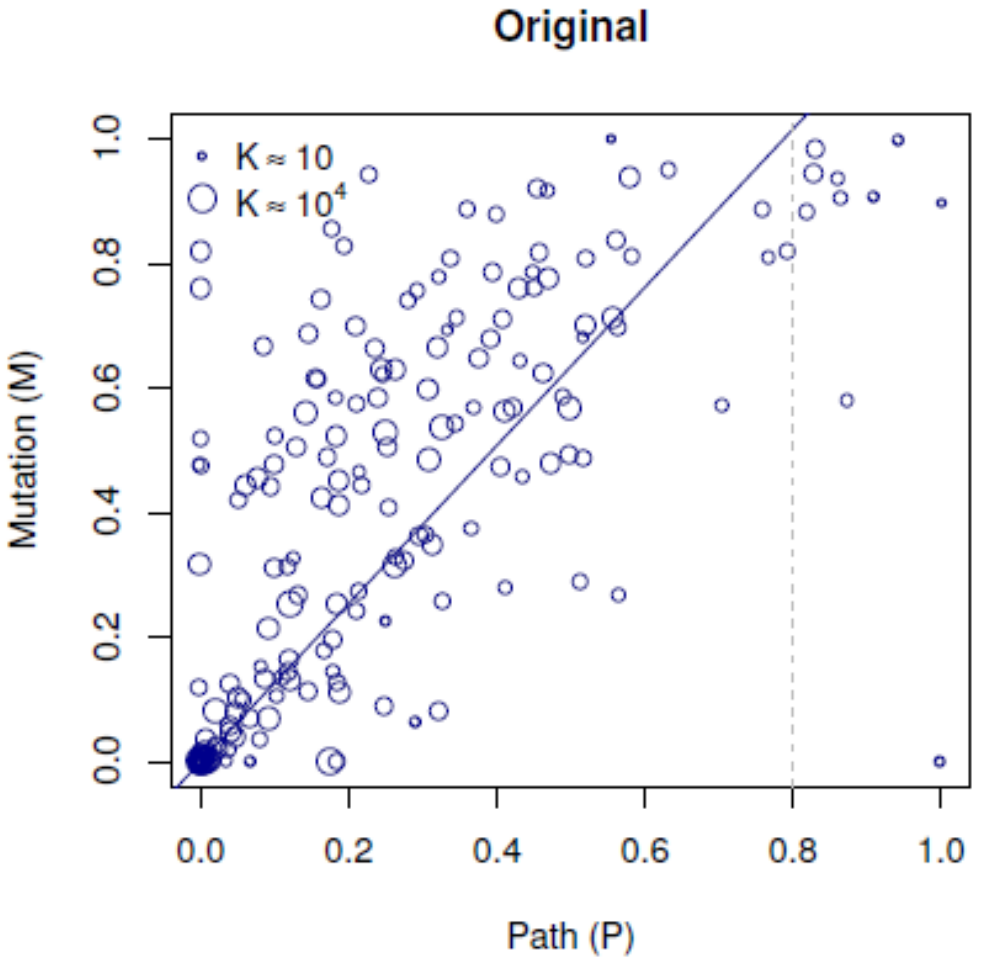

Original
K ≈ 10
K ≈ 10^4
Mutation (M)
Statement (S)
0.0
0.2
0.4
0.6
0.8
1.0

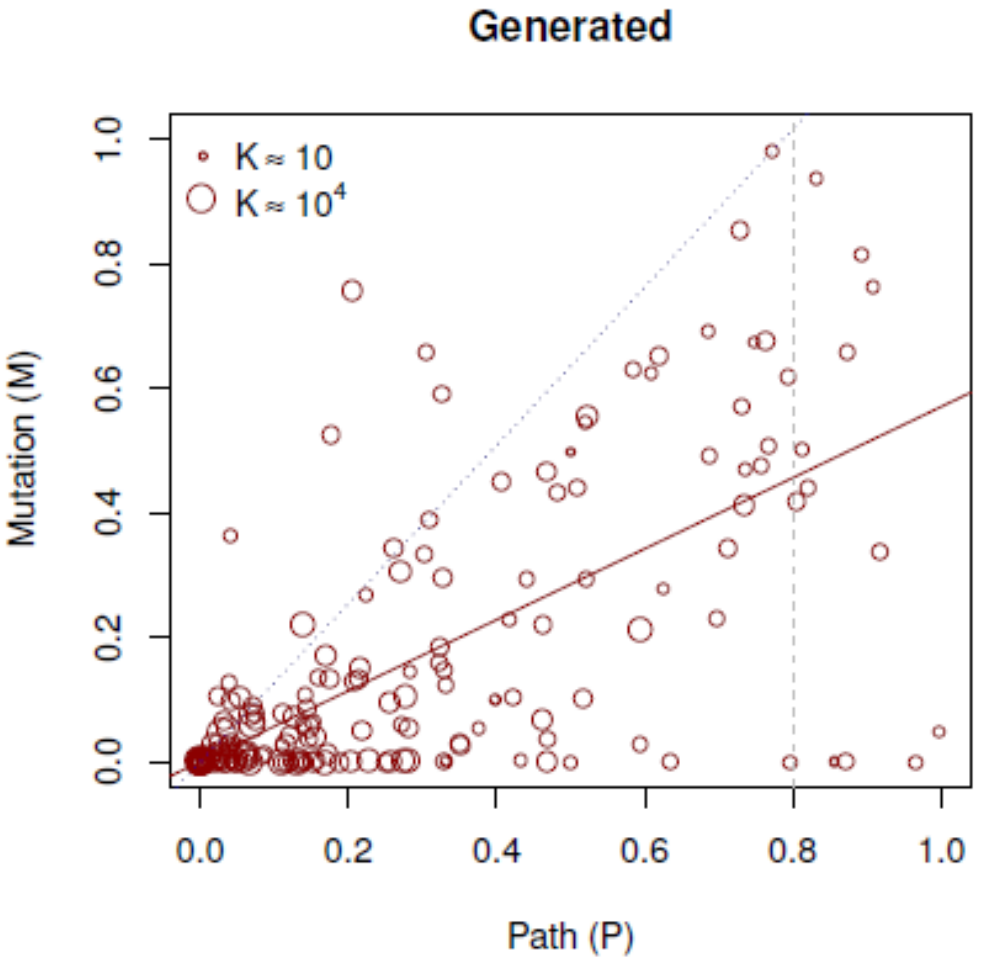

Generated
K ≈ 10
K ≈ 10^4
Mutation (M)
Statement (S)
0.0
0.2
0.4
0.6
0.8
1.0

B

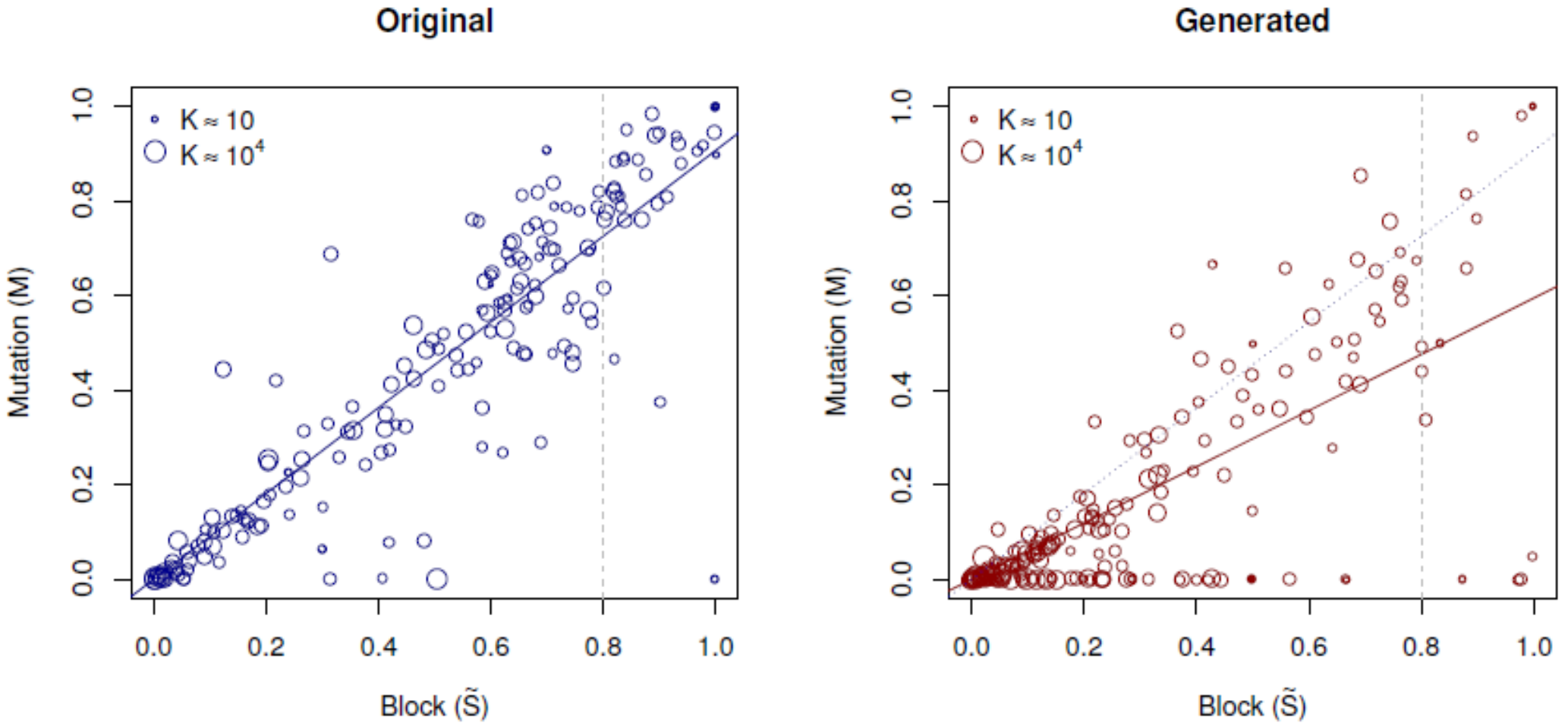

Original
Generated
K ≈ 10
K ≈ 10^4
Mutation (M)
Branch (B)
0.0
0.2
0.4
0.6
0.8
1.0

B


Original
K ≈ 10
K ≈ 10^4
Mutation (M)
Path (P)
0.0
0.2
0.4
0.6
0.8
1.0


Generated
K ≈ 10
K ≈ 10^4
Mutation (M)
Path (P)
0.0
0.2
0.4
0.6
0.8
1.0

B


Original
Generated
K ≈ 10
K ≈ 10^4
Mutation (M)
Block (S̃)
0.0
0.2
0.4
0.6
0.8
1.0

Statistische Methoden

- Kendall Korrelation
  - (Abdeckung, Ergebis)
    Abdeckung1 < Abdeckung2 && Ergebnis1 < Ergebnis2
  - Rate: # (un)vergleichbaren Tupeln durch # aller Tupeln
- R^2 Korrelation / Lineare Regression
  - Linear steigende Abdeckung –!→ linear Steigende Leistung
  - Wenn A x% mehr Abdeckung als B hat,
    ist es auch um c*x% besser?

Bestes Abdeckungskriterium

A

Branch Coverage (BC)

B

Statement Coverage (SC)

Quellenangaben

- **Code Coverage for Suite Evaluation by Developers**
  Rahul Gopinath, Carlos Jensen, Alex Groce
- **Comparing Non-adequate Test Suites using Coverage Criteria**
  Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, Darko Marinov
- **Softwar-Qualität - Testen, Analysieren und Verifi zieren von Software**
  Peter Liggesmeyer
- **https://swt.cs.tu-berlin.de/lehre/seminar/ ss03/folien/ codeabdeckung.pdf**
  25.12.2014, 16.40 Uhr
- **http://de.wikipedia.org/wiki/Kontrollflussorientierte_Testverf ahren**
  19.12.2014, 10.20 Uhr

Quellenangaben

- **http://de.wikipedia.org/wiki/Dynamisches_Software-Testverfahren**
  11.12.2014, 22.53 Uhr
- **Efficient Path Profiling**
  Thomas Ball, James R. Larus
- **http://www.bullseye.com/coverage.html**
  09.12.2014, 17.23 Uhr
- **A Theory of Predicate-Complete Test Coverage and Generation**
  Thomas Ball
- **http://www.atollic.com/index.php/trueanalyzer/types-of-code-coverage-analysis**
  30.12.2014, 14.10 Uhr

Quellenangaben

- **http://www11.informatik.uni-erlangen.de/Lehre/WS0809/PR-SWE/Folien/fa10.pdf**
  10.12.2014, 20.37 Uhr
- **http://www.pst.informatik.uni-muenchen.de/lehre/WS0203/oose/vorlesung_qs.pdf**
  02.01.2015, 20.00 Uhr