EINFLUSS DER PROGRAMMIERERFAHRUNG AUF DEN PROZESS DER TESTGETRIEBENEN ENTWICKLUNG

Andreas Höfer · Dr. Matthias M. Müller

Testgetriebene Entwicklung (TGE)

- TGE führte zunächst in der experimentellen Softwaretechnik ein Schattendasein.
- Ab 2001 befassten sich Studien mit dieser Art der Softwareentwicklung.
- Mittlerweile existiert eine Fülle von Studien mit unterschiedlichen Vergleichen.
- Schwerpunkt der Studien liegt im Vergleich der Qualität der erstellten Software.

Merkmale der Entwicklungstechniken

Entwicklungstechnik	Testautomatisierung	Testzeitpunkt	Granularität
Testgetriebene Entwicklung	Ja	Vorher	Methode
Iterative Test-danach Entwicklung	Ja	Nachher	Methode
Regressionstest	Ja	Nachher	Modul
Konventionelle Entwicklung	Nein	Nachher	Modul

Bisherige Studien

Studie	Teilnehmer	Produktivität	Qualität
TGE vs. Regressionstest			
Müller & Hagner 2002	Studenten	schlechter	kein Unterschied
TGE vs. Iterative Test-danach Entwicklung			
Pančur et al. 2003	Studenten	kein Unterschied	keine Angabe
Geras et al. 2004	Entwickler	kein Unterschied	keine Angabe
Erdogmus et al. 2005	Studenten	besser	kein Unterschied
TGE vs. Konventionelle Entwicklung			
Edwards 2003	Studenten	keine Angabe	besser
George & Williams 2003	Entwickler	schlechter	besser

Aber…

- TGE Geflecht aus unterschiedlichsten Methoden
- Anwendung erfordert Programmiererfahrung
- Lernprozess oft langwierig
- Teilnehmer der Studien sind oftmals Studenten

Wie belastbar sind die Ergebnisse?

- Folgten die Teilnehmer dem vorgeschriebenen Entwicklungsprozess?
- Wenn ja, wie weit hielten sie sich an die Regeln?
- Macht es einen Unterschied ob erfahrene oder unerfahrene Entwickler eingesetzt wurden?

Ziel der Studie

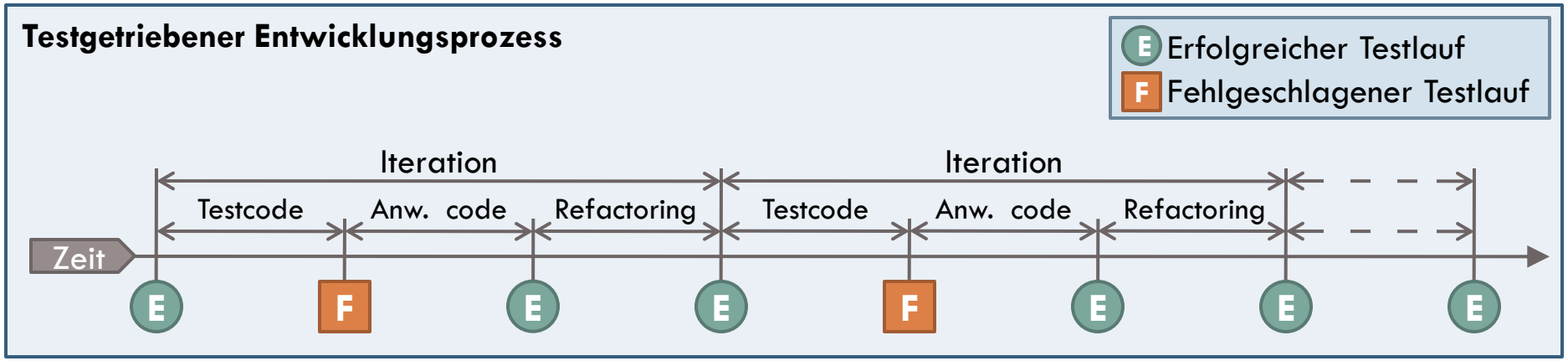
- Zentrale Frage:
  Setzen erfahrene Entwickler die TGE anders um als unerfahrene Entwickler?
- Wenn ja: Sind die bisherigen Ergebnisse das Resultat der vorgesehen Entwicklungstechniken?
- Außerdem: Wie sieht der Entwicklungsprozess der TGE aus?

Die Studie

Die Studie: Übersicht

- Beobachte das Verhalten von…
  - in der TGE erfahrenen Entwicklern
  - TGE Anfängern
- Vergleiche das Verhalten der beiden Gruppen

Schema TGE


Testgetriebener Entwicklungsprozess
E Erfolgreicher Testlauf
F Fehlgeschlagener Testlauf
Iteration
Iteration
Testcode
Anw. code
Refactoring
Testcode
Anw. code
Refactoring
Zeit
E
F
E
E
F
E
E
E

Regeln der TGE

Regel 1 Eine Änderung im Anwendungscode ist nur in einer Methode erlaubt, die zuvor von einem fehlgeschlagenen Testfall aufgerufen wurde.

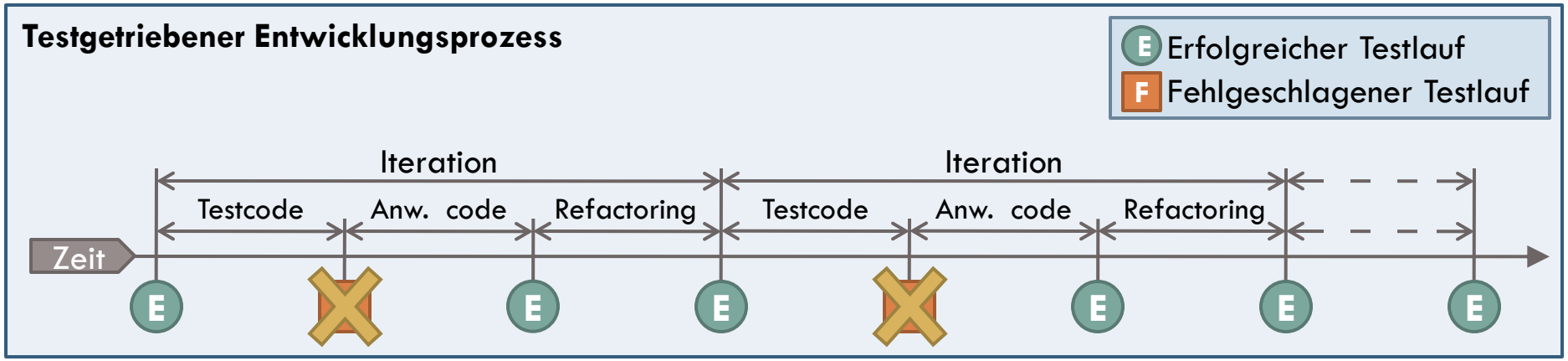
Regel 2 Eine neue Methode muss später von einem fehlschlagenden Testfall aufgerufen werden.

Regel 3 Eine Umstrukturierung ändert die Struktur aber nicht das Verhalten des Programmtextes.

(abgeleitet aus Beck 2000 und Fowler 1999)

Variante von Regel 1: Schwache TGE

- Kein fehlgeschlagener Testfall zwischen Test- und Anwendungscodeänderung



Die Teilnehmer: Entwickler

- 7 Entwickler, durchschnittlich
  - 7,3 Jahre industrielle Programmiererfahrung
  - 3,4 Jahre Erfahrung mit TGE
  - 6,2 Jahre Erfahrung mit Java
- Für Teilnahme bezahlt
- Bearbeitung der Aufgabe an unterschiedlichen Terminen

Die Teilnehmer: Studenten

- 18 Studenten, durchschnittlich
  - im 8. Semester
  - 6 Jahre Programmiererfahrung
  - 2,4 Jahre Erfahrung mit Java
- Teilnehmer des XP-Praktikums
- 11Studenten in der Auswertung
- Gleichzeitige Bearbeitung der Aufgabe

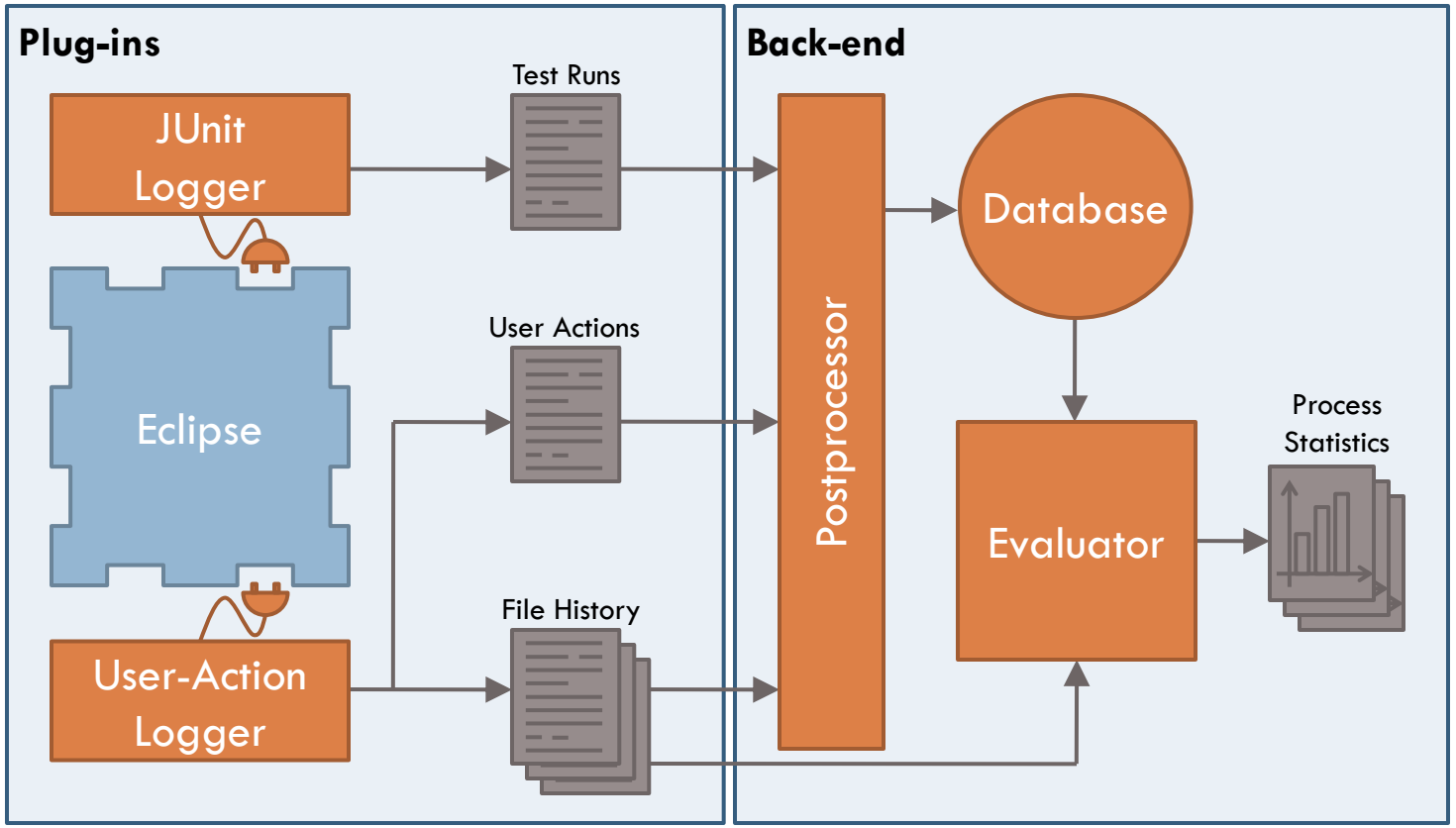
Methodisches zur Datensammlung

- Muss für die Teilnehmer transparent sein, sonst keine verlässlichen Daten
- Prozessdaten sind persönliche Daten
- Zustimmung der Teilnehmer erforderlich
- Jeder Teilnehmer erklärte sich schriftlich mit Datensammlung einverstanden

Welche Daten müssen erhoben werden?

- Kopien aller Quelldateien bei jedem Übersetzen (und Speichern)
- Zeitpunkte der Testläufe und Liste der darin ausgeführten Testfälle
- Aus den vorhergehenden zwei Punkten folgen:
  - Zustand des Programms zu jedem Testlauf
  - Änderungen am Test- und Anwendungscode
- Bearbeitungsdauer von Test- und Anwendungscodeänderungen

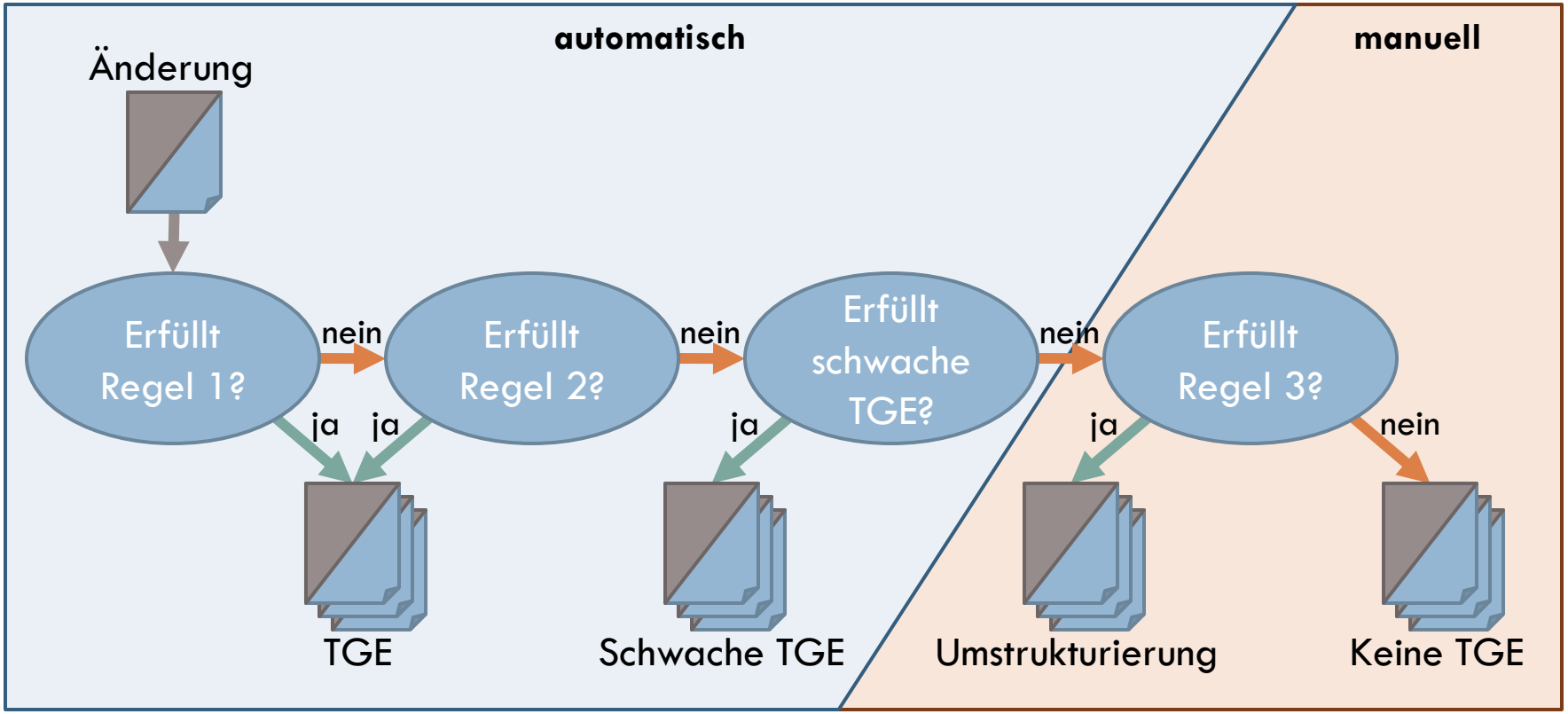
Das System


Plug-ins
Back-end
JUnit Logger
Eclipse
User-Action Logger
Test Runs
User Actions
File History
Postprocessor
Database
Evaluator
Process Statistics

Vorgehen bei der Auswertung

- Betrachte für jeden Teilnehmer jede Änderung am Programmtext.
- Klassifiziere jede Änderung in die folgenden Kategorien:
  - TGE
  - Schwache TGE
  - Umstrukturierung
  - Keine TGE

Klassifikation der Änderungen



Automatische Klassifikation gemäß Regel 1

Prüfe für jede Änderung in Zeile l der Datei f zum Zeitpunkt t ob sie den Regeln der TGE genügt.

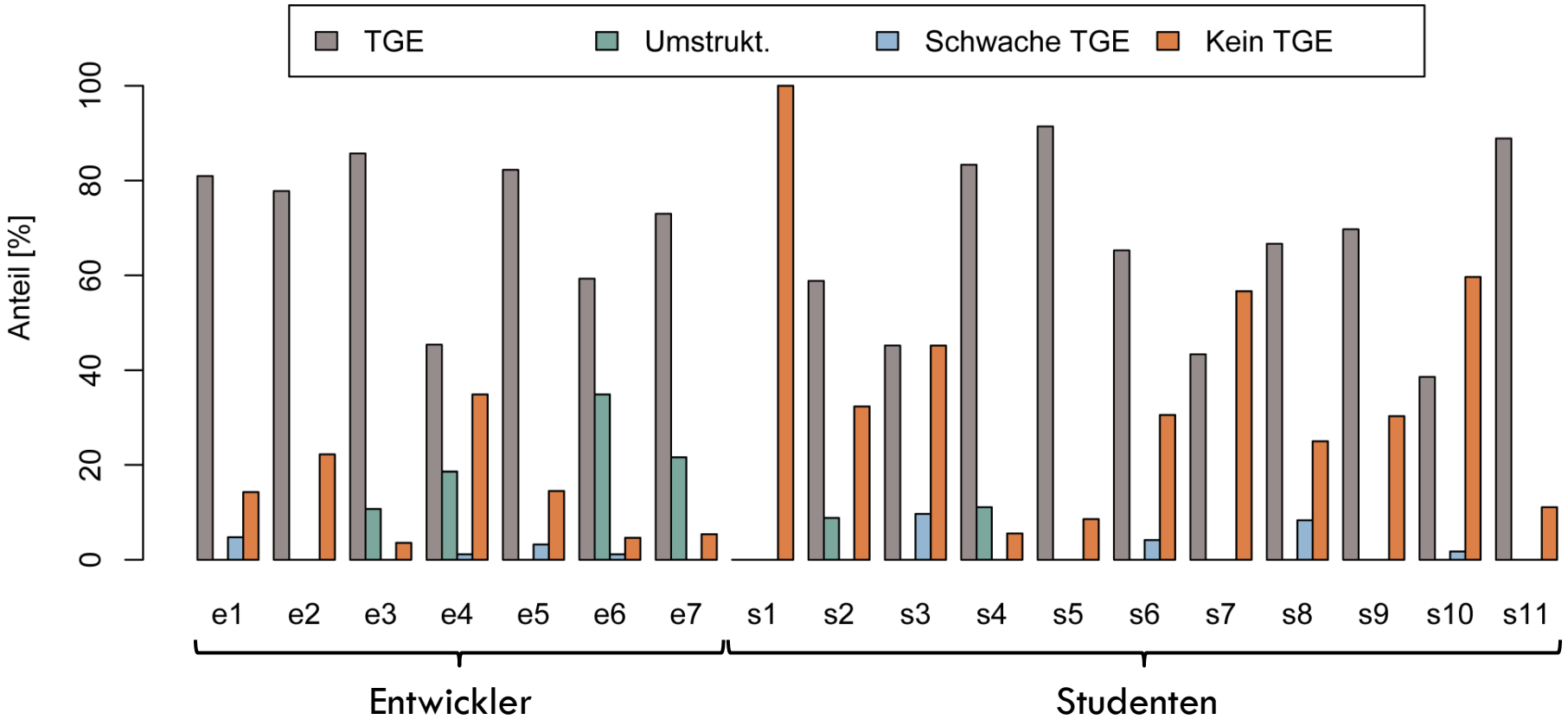
```
def isTDD(Line l, File f, Time t):
  if isApplicationFile(f):
    Method m = getMethod(l, f, t)
    TestRun tr = getPreviousTestRun(t)
    if tr != null:
      Time tprev = tr.getTime()
      Methods tm = getTestMethodsByCallee(tprev, m)
      if tm != null:
        return failed(tm, tprev)
  return false
```

Manuelle Klassifikation

- Definition von Änderungsmustern
- Einteilen aller verbliebenen Änderungen in diese Änderungsmuster
- Einteilung von uns unabhängig voneinander durchgeführt
- Abweichungen wurden aufgelöst

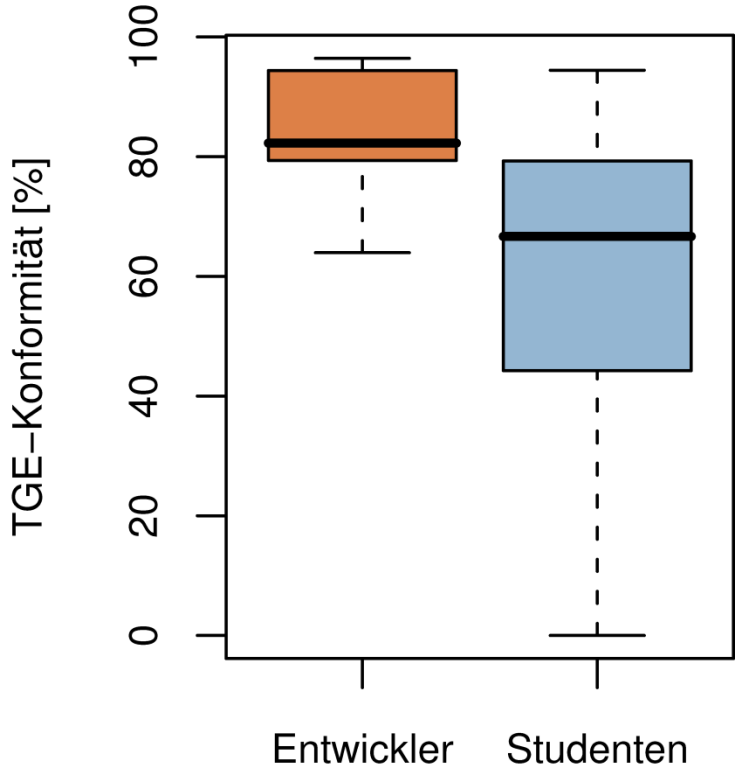
Die Ergebnisse

Änderungsverhalten der Teilnehmer



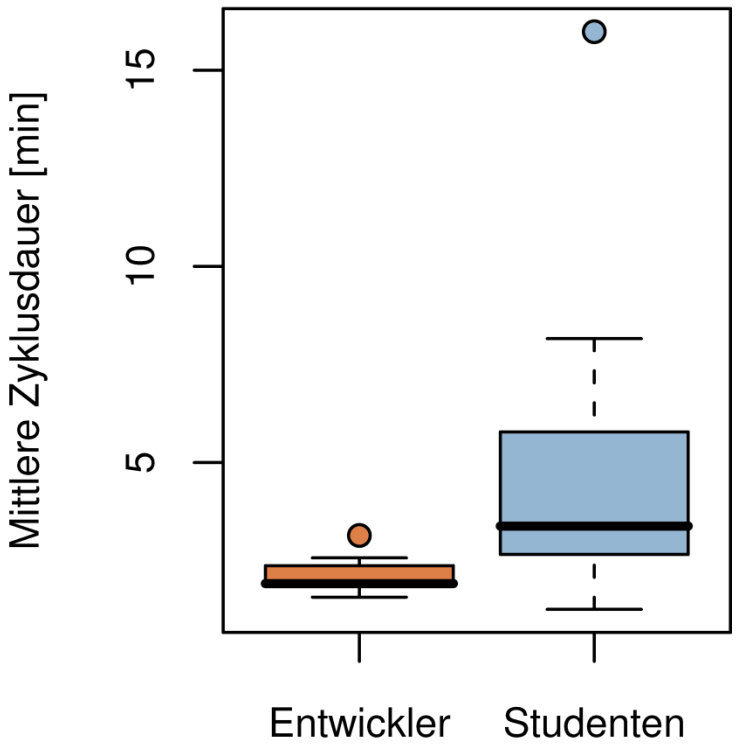
TGE-Konformität

$$\text{TGE} - \text{Konformität} = \frac{|\text{TGE}| + |\text{Umstrukturierung}|}{|\text{alle Änderungen}|}$$



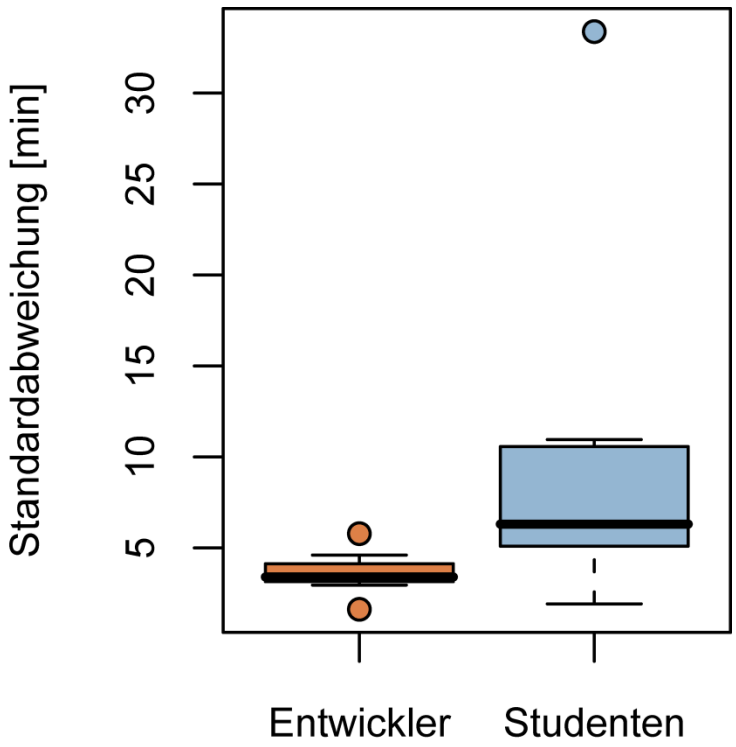
- Entwickler halten sich mehr an die Regeln der TGE.
- Studenten weisen größere Streuung auf.
- Manche Studenten sind mit Entwicklern vergleichbar.
- Nur wenige Entwickler erreichen mehr als 90%.

Zyklusdauer



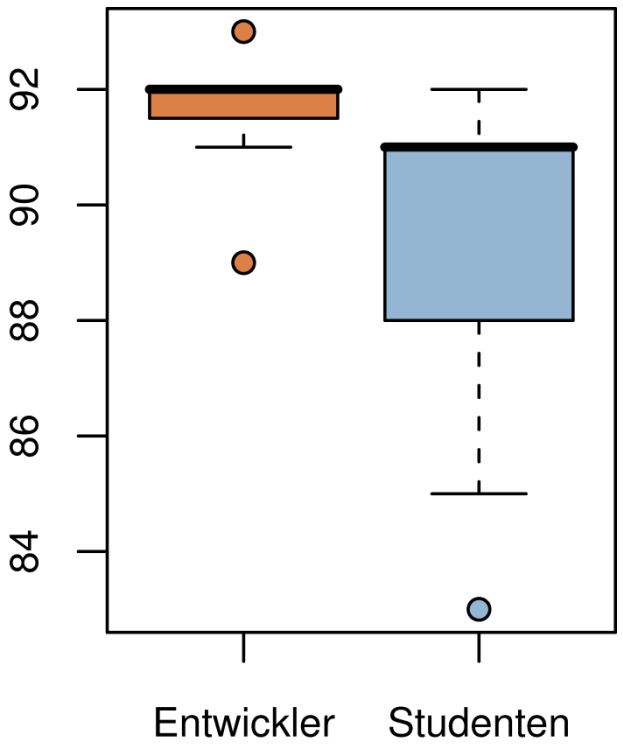
- Die TGE Prozesse der Entwickler haben eine kürzere Zyklusdauer.
- Starke Schwankungen bei den TGE Prozessen der Studenten.

Standardabweichung der Zyklusdauer



- Entwickler arbeiten konstanter.
- Fazit Zyklusdauer: Das TGE Herz der Entwickler schlägt schneller und konstanter.

Testabdeckung (Grundblöcke)



- Entwickler erreichten eine höhere Test-abdeckung.
- Sie änderten den Test-code im kleineren Umfang als die Studenten.
- Ein ähnliches Bild ergibt sich für die Zeilenabdeckung.

Gültigkeit der Studie

- Gibt es, abgesehen von der TGE-Erfahrung, andere Erklärungen für die Ergebnisse?
- Die Studenten hatten weniger Programmier-erfahrung, es wurden schon Studenten von der Auswertung ausgeschlossen.
- Der tägliche TGE Prozess könnte vom gemessenen Prozess abweichen: Die Teilnehmer wussten, dass sie beobachtet werden.
- Unterschiedliche Motivation: Die Entwickler wurden bezahlt, die Studenten nicht.

Fazit

- Prozesse der Entwickler und Studenten unterschieden sich in fast allen untersuchten Kenngrößen.
- Gültigkeit der bisherigen Studien scheint somit fragwürdig: Welchem Prozess folgten die Teilnehmer?
- Ab wann gilt eine Entwicklungsmethode testgetrieben: Ab 100%, 80% oder 60%?
- Hat die Paarprogrammierung einen Einfluss auf die Einhaltung der Regeln der TGE?

Sponsoren


Deutsche
Forschungsgemeinschaft
DFG


andrena
OBJECTS


e-tecture

Werbung

- Wir suchen professionelle Entwickler für weitere Experimente!
- Haben Sie mind. 1 Jahr industrielle Programmier-erfahrung mit den folgenden Techniken?
  - Java
  - TGE
  - Paarprogrammierung
- Dann bewerben Sie sich noch heute!
- Kontakt: Andreas Höfer
  http://www.ipd.uka.de/Tichy/people.php?id=30

Referenzen

Beck 2000 Beck, K. *Extreme Programming Explained: Embrace Change.* Addison-Wesley, 2000

Edwards 2003 Edwards, S. H. *Using Test-Driven Development in the Classroom: Providing Students with Automatic, Concrete Feedback on Performance.* Proceedings of the International Conference on Education and Information Systems: Technologies and Applications (EISTA'03), International Institute of Informatics and Systemics, 2003, S. 421-426

Erdogmus et al. 2005 Erdogmus, H.; Morisio, M.; Torchiano, M. *On the Effectiveness of the Test-First Approach to Programming.* IEEE Transactions on Software Engineering, 2005, 31(3), S. 226-237

Fowler 1999 Fowler, M. *Refactoring – Improving the Design of Existing Code.* Addison-Wesley, 1999

George & Williams 2003 George, B.; Williams, L. *An Initial Investigation of Test Driven Development in Industry.* SAC '03: Proceedings of the 2003 ACM symposium on Applied computing, ACM Press, 2003, S. 1135-1139

Referenzen

Geras et al. 2004 Geras, A.; Smith, M.; Miller, J.
A Prototype Empirical Evaluation of Test Driven Development. METRICS~'04: Proceedings of the 10th International Symposium on Software Metrics, IEEE Computer Society, 2004, S. 405-416

Müller & Hagner 2002 Müller, M. M.; Hagner, O.
Experiment about Test-First Programming.
IEE Proceedings – Software, 2002, 149(5), S. 131-136

Müller & Höfer 2007 Müller, M. M.; Höfer, A.
The Effect of Experience on the Test-Driven Development Process. Erscheint in Empirical Software Engineering.

Pančur et al. 2003 Pančur, M.; Ciglarič, M.; Trampuš, M. & Vidmar, T.
Towards Empirical Evaluation of Test-Driven Development in a University Environment. EUROCON 2003; Computer as a Tool; The IEEE Region 8, 2003, Band 2, S. 83-86