Behandlung von Nebenläufigkeitsaspekten in einem Werkzeug zur Verteilten Paarprogrammierung

Sebastian Ziller
Institut für Informatik
FU Berlin
29.10.09

Inhalt

- Verteilte Paarprogrammierung
- Saros – ein Werkzeug zur Verteilten Paarprogrammierung
- Mehrschreibermodus in Saros
- Probleme im Mehrschreibermodus:
  - Undo-Funktionalität
  - Synchronisierung von komplexen Operationen
- Empirische Studie

Verteilte Paarprogrammierung

Freie Universität Berlin

Paarprogrammierung:

- Zwei Entwickler, ein Rechner
- Vorteile: höhere Disziplin, weniger Defekte, Wissenstransfer...

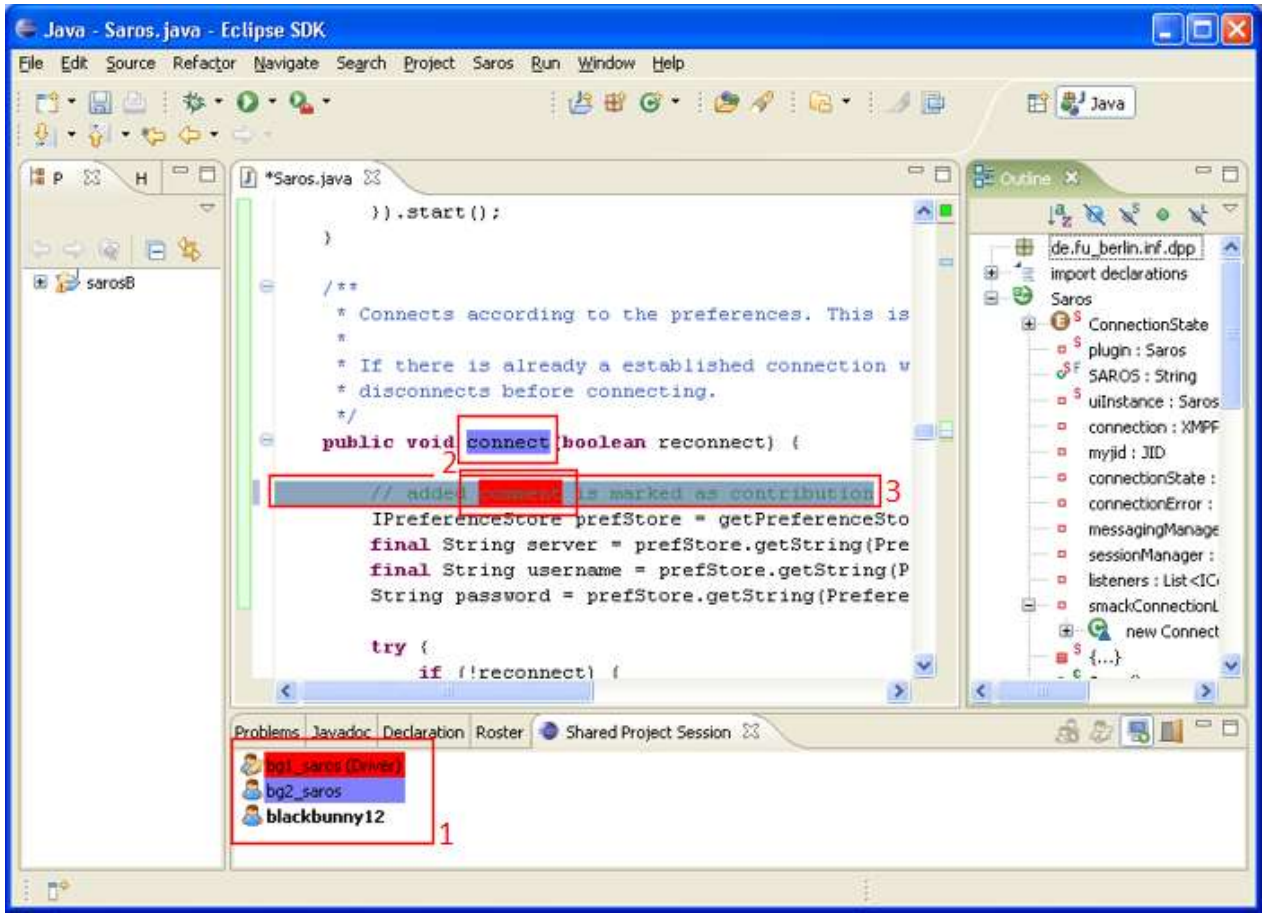
Verteilte Paarprogrammierung:

- Zwei Entwickler, ein Artefakt, aber zwei verschiedene Arbeitsplätze
- Vorteile: durch Globalisierung nicht immer reguläre Paarprogrammierung möglich

Freie Universität Berlin

Saros

- Werkzeug zur Verteilten Paarprogrammierung
- Eclipse-Plugin
- Bis zu fünf Teilnehmer
- Mehr-Schreiber-Funktionalität



Mehrschreibermodus in Saros

Freie Universität Berlin

- Teilnehmer haben eine der beiden Rollen
  - Driver
  - Observer
- Mehr als ein Driver möglich
- Mehrere Teilnehmer können parallel ein Dokument bearbeiten
- Nebenläufigkeitskontrolle durch Jupiter und GOTO Inclusion Transformation

Anforderungen an Undo

- im Ein-Benutzer-Betrieb:

letzte Operation rückgängig machen

- im Mehr-Benutzer-Betrieb:

letzte **eigene** Operation rückgängig machen

Probleme

- Was war meine letzte eigene Operation?

- Wie mache ich sie rückgängig?

z. B. Text: „abc“
lokale Operation: einfüge (3, „def“) → „abcdef“
viele entfernte Operationen → „uvwxyzabcd1eklm“
undo(einfüge (3, „def“)) → ?

Lösung

- Was war meine letzte eigene Operation?

Leicht,
Operationen merken (mit Eigenschaft lokal, entfernt)

einfüge (0,"uvwxyz")	entfernt
lösche (5, "f")	entfernt
einfüge (4, "1")	entfernt
einfüge (3, "def")	lokal

Freie Universität Berlin

Lösung

- Wie mache ich sie rückgängig?

Letzte lokale Operation umkehren und sie gegen jede jüngere Operation transformieren

Was macht die Transformation?

- $T(O_A, O_B) \rightarrow O_A'$
- Anwendung von O_B, dann O_A' bringt erwünschtes Ergebnis

z.B. Text: „abcdef“

- O_A = lösche (3, „def“)
- O_B = einfüge (4, „1“)
- $T(O_A, O_B)$ = Split (lösche (3, „d“), lösche (5, „ef“))

- → GOTO Inclusion Transformation

Lösung

Letzte lokale Operation umkehren und sie gegen jede jüngere Operation transformieren

Komplexe Operationen

- Bisher: primitive Operationen „löschen“ und „einfügen“
- hier: komplexe Operationen wie Konsistenzwiederherstellung und Rollenwechsel

Naive Umsetzung am Beispiel Rollenwechsel

- Gastgeber G möchte einem Driver D das Schreibrecht entziehen
- G schickt D eine entsprechende Nachricht und setzt ihn in seiner Nutzerverwaltung auf Observer
- D empfängt die Nachricht und setzt sich in seiner Nutzerverwaltung auf Observer

- Problem: Während die Nachricht unterwegs ist, kann D weitere Textänderungen generieren
- G wendet sie aber nicht an, da er von D keine Änderungen mehr erwartet

Lösung

- G schickt Anfrage an D keine Änderungen mehr zu generieren
- D sperrt daraufhin seine Editoren und schickt Bestätigung an G
- G empfängt Bestätigung setzt D auf Observer und schickt Nachricht an alle, dass D jetzt Observer ist
- D empfängt Nachricht und setzt sich auf Observer
- G teilt D mit, dass die Sperre aufgehoben werden kann

- Generischer Mechanismus:

Anfrage – Sperren / Entsperren – Bestätigung

Synchronisierung kompl. Operationen

Freie Universität Berlin

Generische Lösung:

Anfrage – Sperren / Entsperren – Bestätigung

Implementiert in Modul StopManager:

- bietet nach außen start- und stop-Methoden, um Teilnehmer zu sperren

- Eingesetzt bei Rollenwechseln, bei Konsistenzwiederherstellungen und im Einladungsprozess

Ziel

- Unterschied zwischen Mehr- und Einzelschreibermodus

- Im Hinblick auf
  - Entwicklungsgeschwindigkeit
    und
  - Codequalität

Ablauf

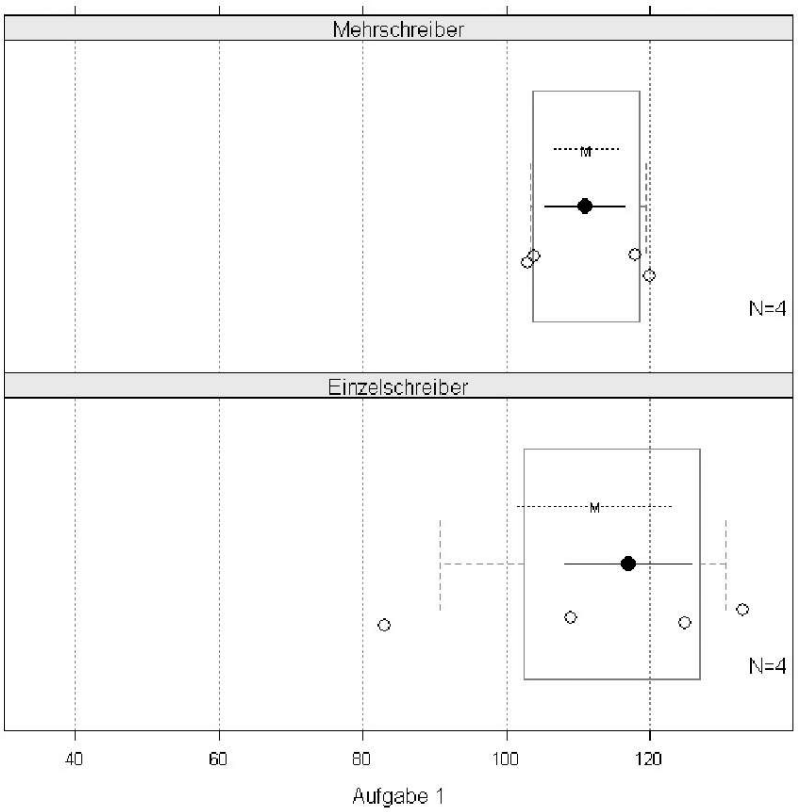
- 16 Teilnehmer (Studenten und junge Absolventen)
- Einteilung in Paare
- Zufällige Einteilung der Paare in Gruppen A und B
- Lösen von zwei Aufgaben 1 und 2

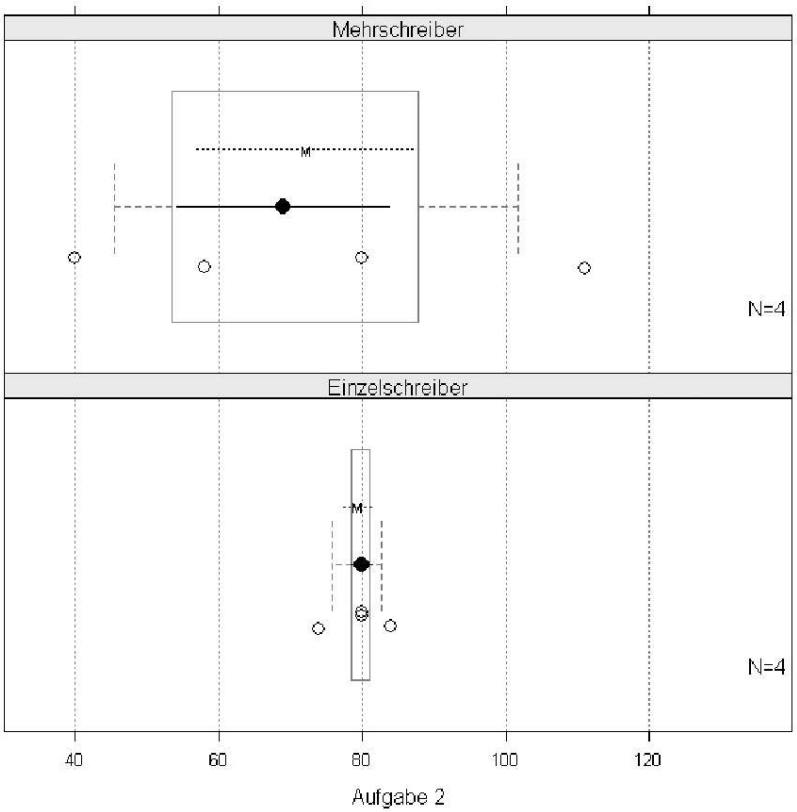
- Gruppe A: Aufgabe 1 im Einzel-, 2 im Mehrschreibermodus
- Gruppe B: Aufgabe 1 im Mehr-, 2 im Einzelschreibermodus

- Messung der Zeit
- Bewertung der Qualität durch Unit-Tests und zyklomatische Komplexität

Freie Universität Berlin

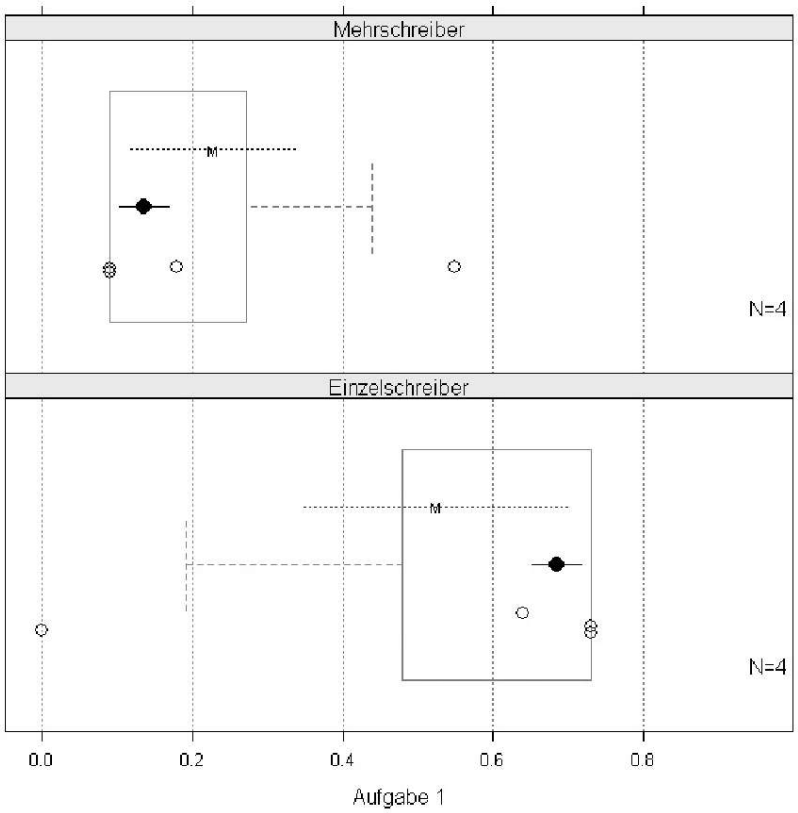
Ergebnisse: Zeitmessung

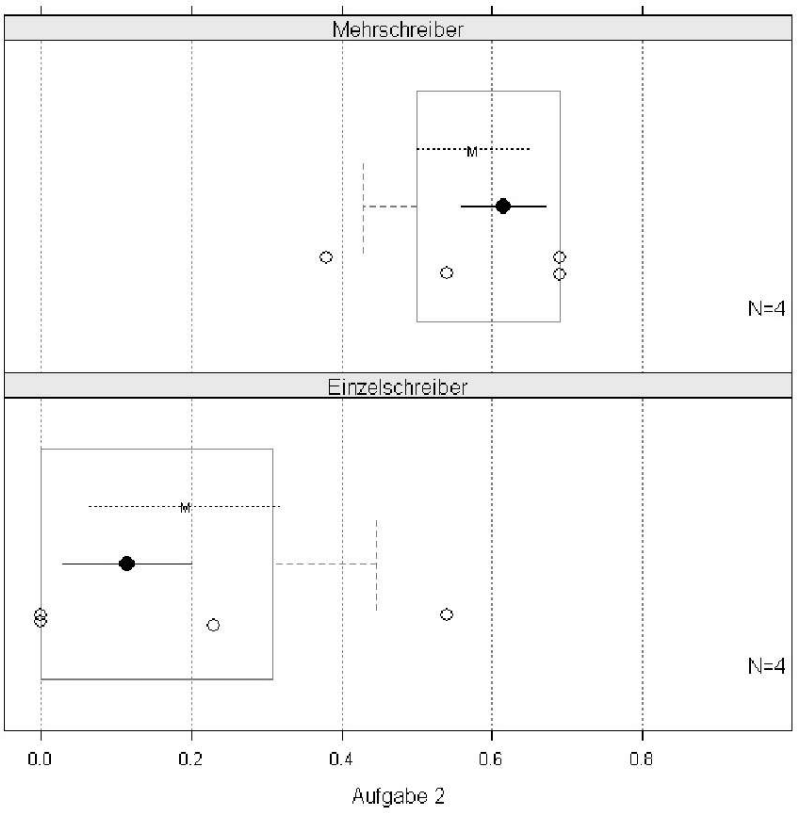




Freie Universität Berlin

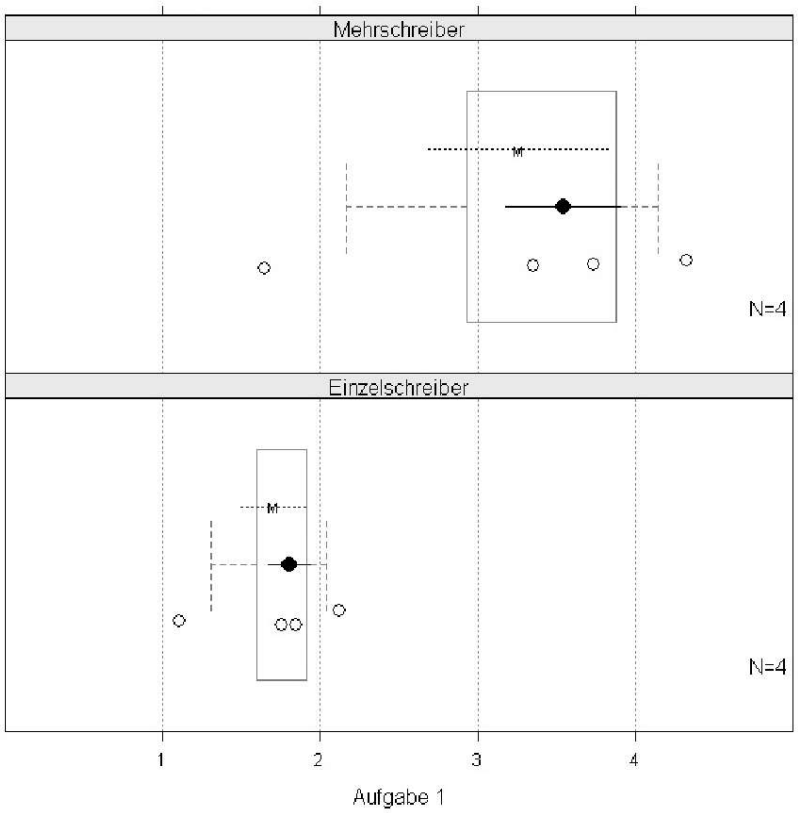
Ergebnisse: Korrektheit (JUnit-Tests)

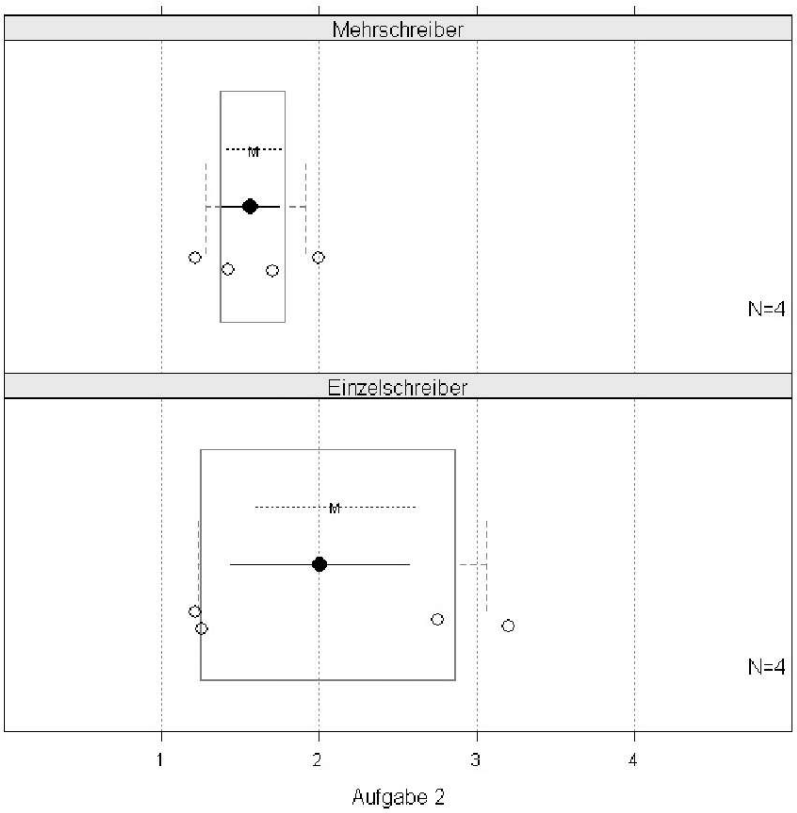




Freie Universität Berlin

Ergebnisse: zyklomatische Komplexität





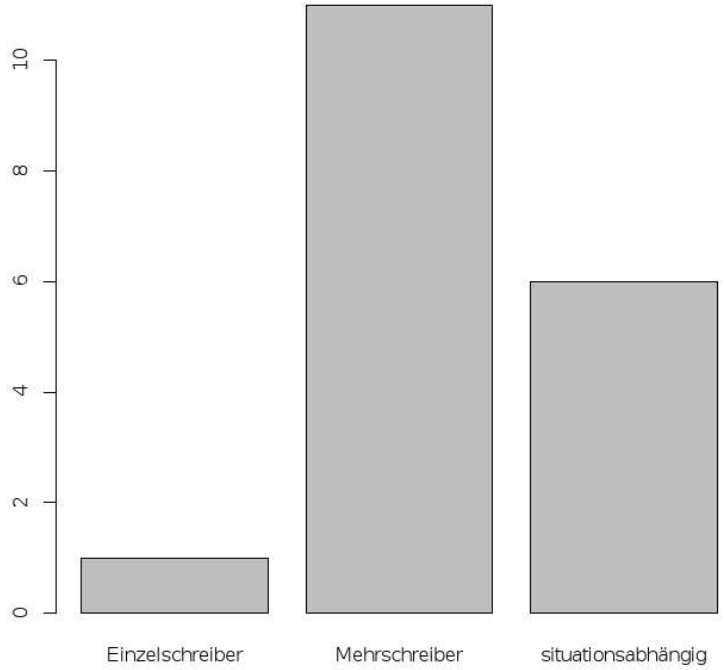
Fazit

- Keine klaren Ergebnisse
- Vermutlich verursacht durch:
  - Wenige Teilnehmer
  - Inkonsistenzen
  - Unterschiedliches Domänenwissen
- Wiederholung wird daher empfohlen
  - Mehr Teilnehmer
  - Stabilere Software

Freie Universität Berlin

Fazit: Mehrschreibermodus erscheint nützlich

Würden Sie bei einer zukünftigen Verwendung den Einzel- oder den Mehrschreibermodus vorziehen?



Vielen Dank!