Konzeptvorstellung Diplomarbeit:

Behandlung von Nebenläufigkeitsaspekten in einem Werkzeug zur verteilten Paarprogrammierung

Sebastian Ziller

Institut für Informatik

FU Berlin

14.05.09

Inhalt

Freie Universität Berlin

- Verteilte Paarprogrammierung
- Saros – ein Werkzeug zur verteilten Paarprogrammierung
- Nebenläufigkeitsprobleme im Mehr-Schreiber-Betrieb
- Lösungsansätze:
  - Jupiter als Nebenläufigkeitskontrollalgorithmus
  - GOTO Inclusion Transformation als Operationstransformationsalgorithmus
  - Undo-Funktionalität
  - Synchronisierung von komplexen Operationen
- Experiment
- Offene Fragen
- Zusammenfassung

Verteilte Paarprogrammierung

Freie Universität Berlin

Paarprogrammierung:

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

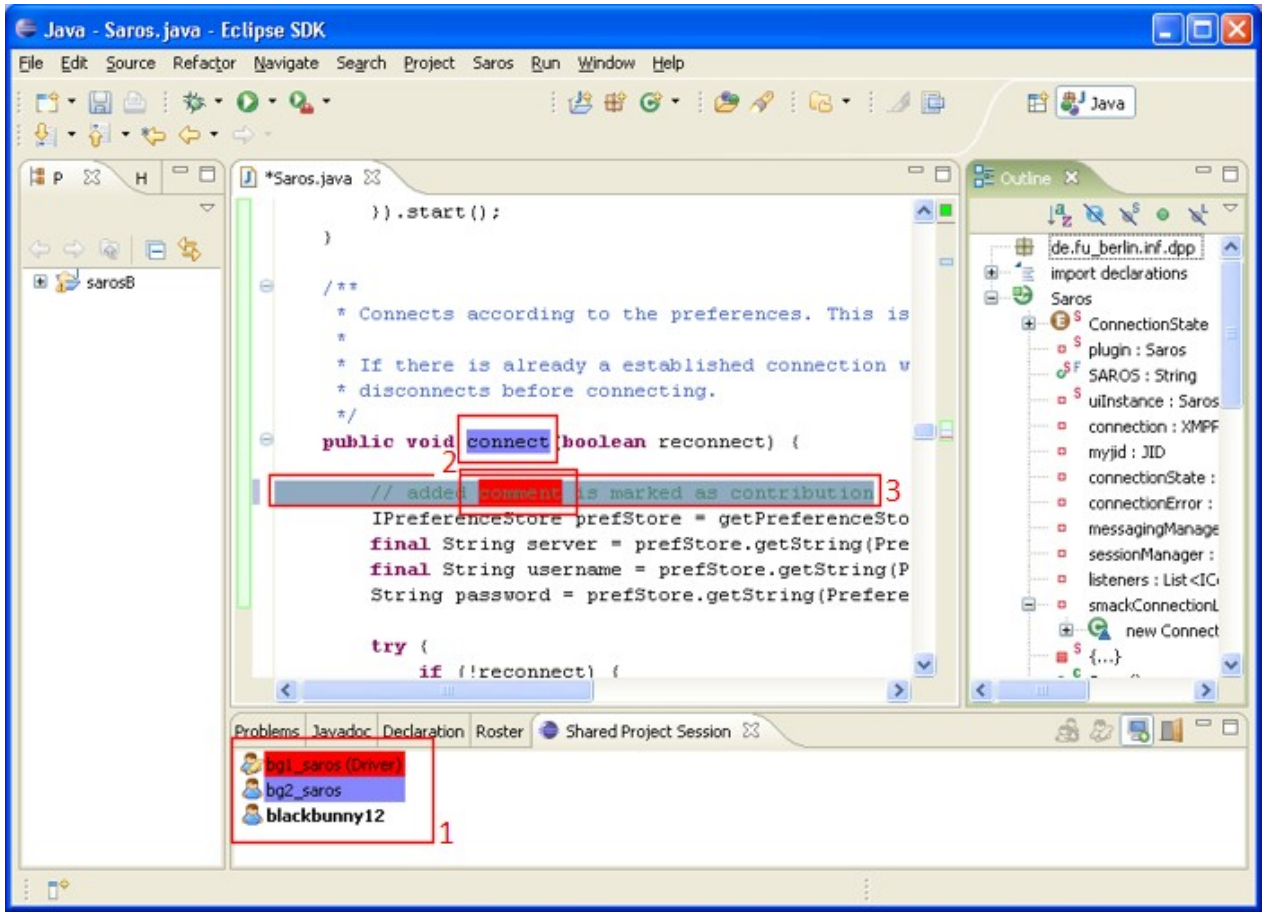
verteilte Paarprogrammierung:

- 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
- Mehr-Schreiber-Funktionalität



Anforderungen an einen Mehr-Schreiber-Betrieb

- Konsistenz:
  - Konvergenz
  - Kausalität
  - Intentionserhaltung
- Geringe Latenzzeit

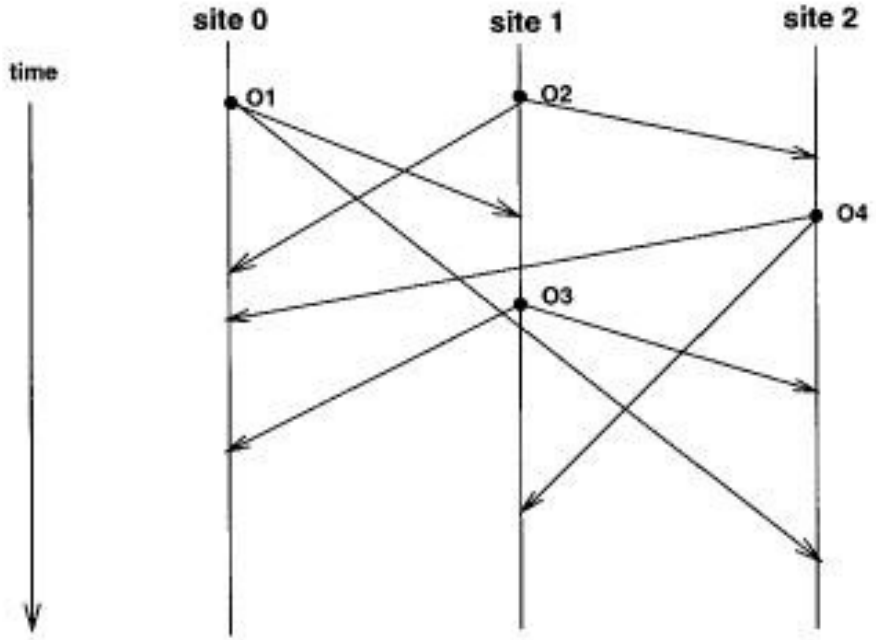
Konvergenz

Jede Seite hat zu jeder Zeit den gleichen Inhalt

temporäre latenzbedingte Divergenzen werden toleriert

Kausalität

- Operationen können voneinander abhängen
- Beachtung der Kausalität heißt Beachtung der Abhängigkeiten
- heißt Beachtung der Reihenfolge



Intentionserhaltung

- Inhalt: „def“
- Entwickler A generiert Einfüge (0, „abc“)
- Entwickler B generiert Einfüge (3, „ghi“)
- Operation A wird angewendet
- Operation B wird angewendet

- Ohne Intentionserhaltung: „abcghidef“
- Mit Intentionserhaltung: „abcdefghi“

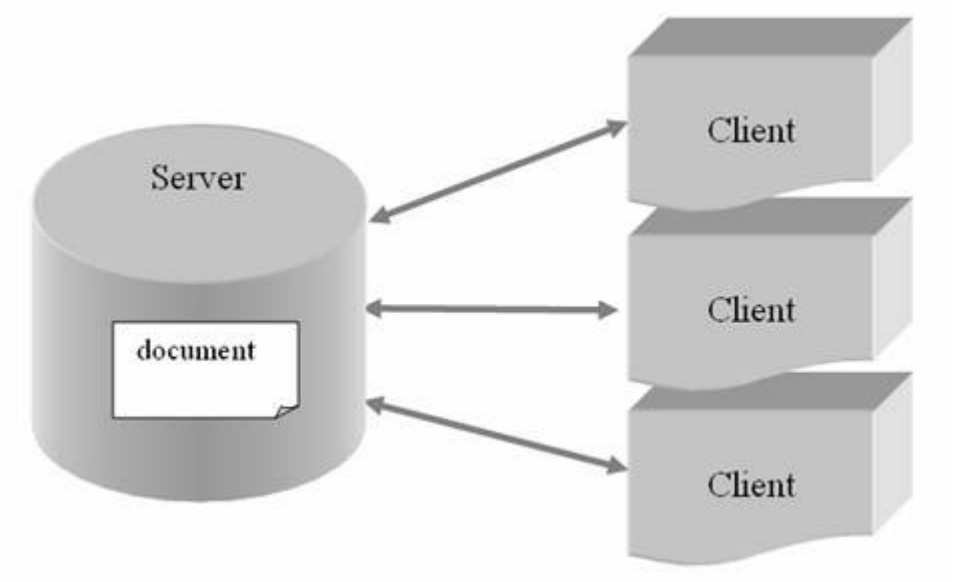
Konvergenz durch zwei verschiedene Ansätze

- Zentrale Architektur
- Replikationsmechanismus

Zentrale Architektur

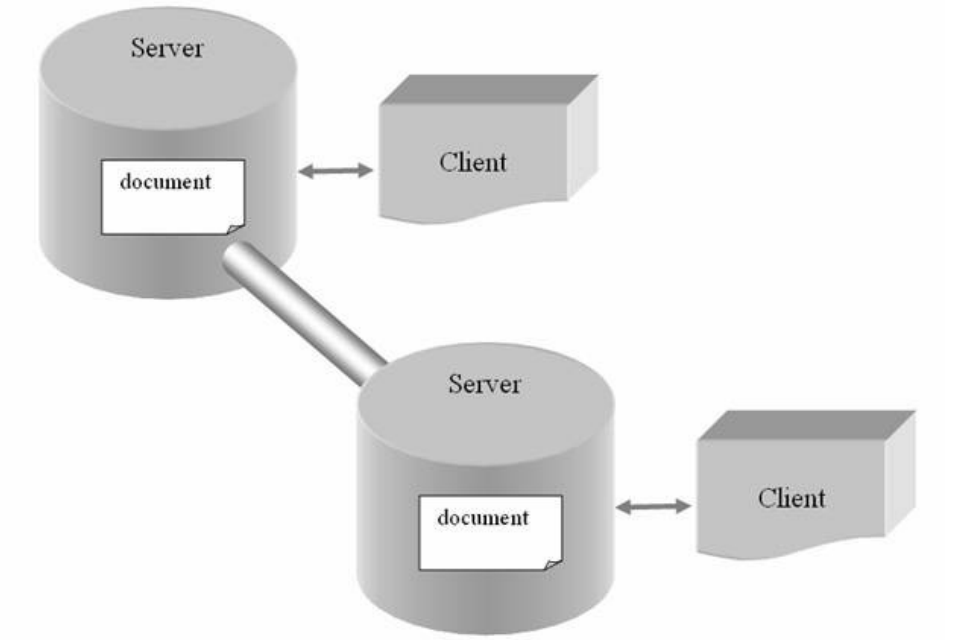
- Ein zentraler Server verwaltet das Dokument
- Er empfängt Änderungen von den Klienten, wendet sie an und sendet den neuen Dokumentstatus zurück

- Nachteil:
  hohe Latenzzeit



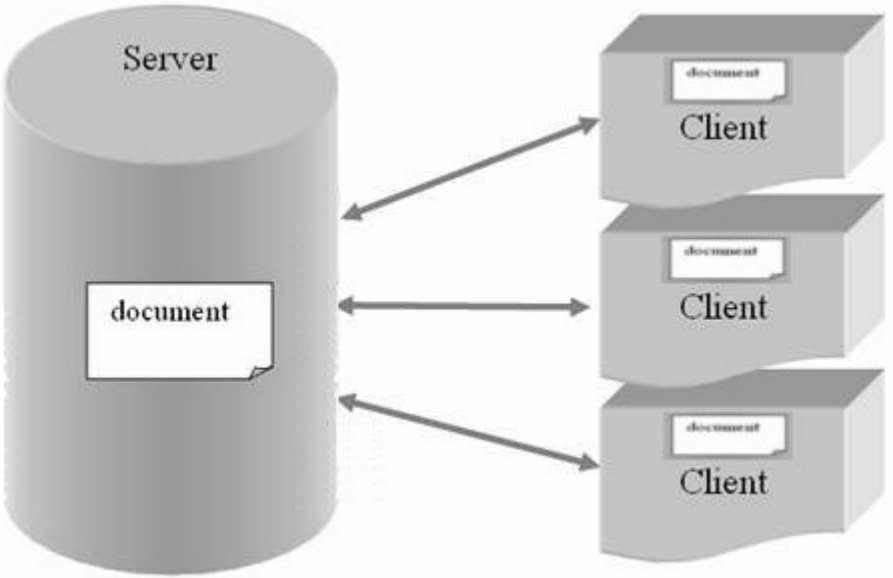
Replikationsmechanismus

- Jeder Klient hat eigenen Server (mit eigenem Dokument) integriert
- Jede Änderung muss an jeden Server kommuniziert und gegen jede Änderung von jeder anderen Seite transformiert werden
- Hoher Transformations-aufwand (1:n-Transformation)



Jupiter

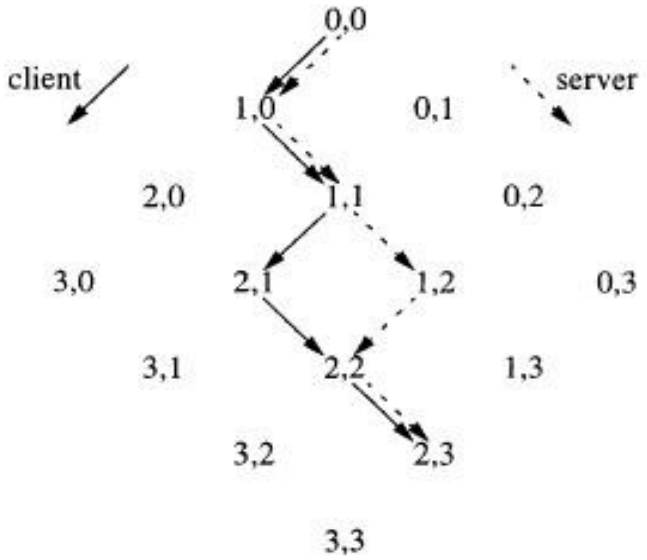
- Kombination aus beiden Ansätzen:
- Ein zentraler Server, aber jeder Klient hat eigenes Dokument
- Änderungen werden lokal sofort angewendet, dann an den Server geschickt



- Geringe Latenzzeit
- 1:1-Transformation

Kausalität und Intentionserhaltung

- Beachtung der Reihenfolge durch Vergabe von Zeitstempeln
- Zeitstempel sind Tupel (lokal, entfernt) von Operationsnummern



Freie Universität Berlin

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 = einfüge(6, „ghi“)
- O_B = lösche(0, „ab“)
- $T(O_A, O_B)$ = einfüge(4, „ghi“)

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

Komplexe Operationen

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

Bisherige Umsetzung am Beispiel Rollenwechsel

- Gastgeber G möchte einem Schreiber S das Schreibrecht entziehen
- G schickt S eine entsprechende Nachricht und setzt ihn in seiner Nutzerverwaltung auf Zuschauer
- S empfängt die Nachricht und setzt sich in seiner Nutzerverwaltung auf Zuschauer

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

Lösung

- G schickt Anfrage auf Rollenwechsel an S
- S sperrt daraufhin seine Editoren und schickt Bestätigung an G
- G empfängt Bestätigung setzt S auf Zuschauer und schickt Rückbestätigung an S
- S empfängt Rückbestätigung und setzt sich auf Zuschauer

- Generischer Mechanismus:

  Anfrage – Sperre – Bestätigung - Rückbestätigung

Synchronisierung kompl. Operationen

Freie Universität Berlin

Generische Lösung:

Anfrage – Sperre – Bestätigung – Rückbestätigung

- Muss noch angepasst werden auf alle Typen komplexer Operationen
- Das sind: Rollenwechsel, Dateien umbenennen / verschieben, Dateien löschen, neu hinzukommende Teilnehmer, Verlassen der Sitzung, Konsistenzwiederherstellung

- Dabei noch viele Details unklar:
  - Umgang mit Langläufern
  - Vergabe von Löschrechten

Ablauf

- Nach Umsetzung der angesprochenen Punkte
- Mind. 12 Teilnehmer, Teams aus je 2 (wahrscheinlich studentischen) Entwicklern
- Sitzung von etwa 3-4 Stunden mit definierten Aufgaben

- Zwei Gruppen: mit Mehrschreibermöglichkeit und ohne

Ziel

- Vergleich von verteilter Entwicklung im Ein-Schreiber-Betrieb und im Mehr-Schreiber-Betrieb

Fragen

- Mit welcher Praktik lassen sich die Aufgaben schneller lösen?
  → Effizienz

- Mit welcher Praktik entsteht der bessere Code?
  → Effektivität

Offene Fragen

- Messung der Codequalität
  - Defektarmut klar, aber sonst...
  - Expertenurteil?
- Vergleichbarkeit der beiden Gruppen
- Aufgabe

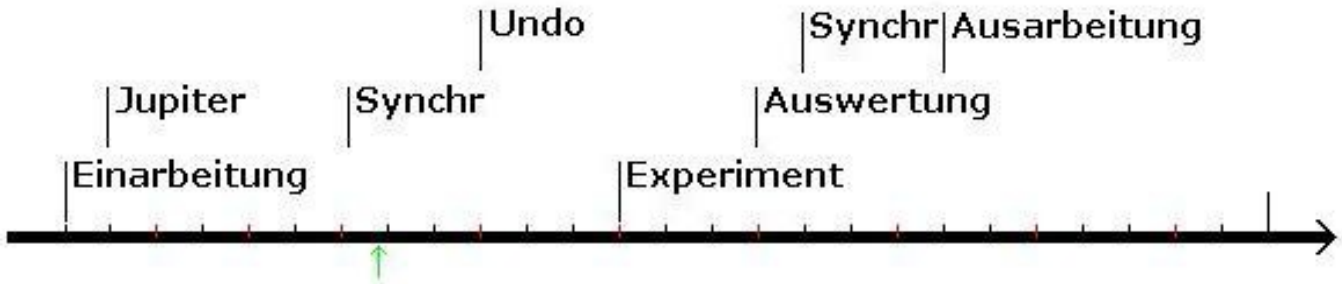
Plan

- Anpassung von Jupiter und Operationstransformationen
- Umsetzung einer Undo-Funktionalität
- Konzeptionierung und Umsetzung einer Synchronisierung von komplexen Operationen
- Experiment zur Bewertung der Mehr-Schreiber-Möglichkeit

Freie Universität Berlin

Zeitliche Planung

- Jupiter und Operationstransformationen (6 Wochen)
- Undo-Funktionalität (3 Wochen)
- Synchronisierung von komplexen Operationen (6 Wochen)
- Experiment (3 Wochen, ab Ende Juni)
- Auswertung (1 Woche)
- Ausarbeitung (6 Wochen)



Vielen Dank!