Freie Universität Berlin

Diplomarbeit

Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung von verteilter Paarprogrammierung

Diplomand: Riad Djemili

Betreuer: Christopher Özbek, Stephan Salinger

Gutachter: Prof. Lutz Prechelt, Prof. Peter Löhr

Institut für Informatik, FU Berlin

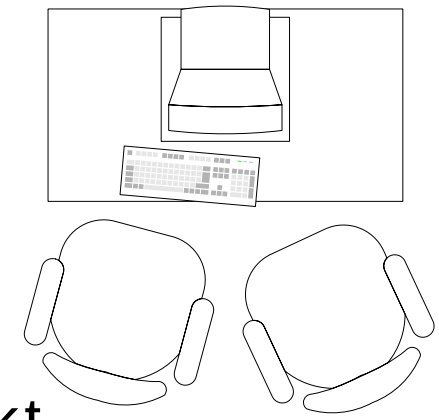
5.10.2006

Übersicht

- Paarprogrammierung
  - Definition
  - Vor-/Nachteile
- Verteilte Paarprogrammierung
  - Definition
  - Vor-/Nachteile
  - Technische Werkzeugansätze
- Umfrage
  - Ergebnisse
- Saros – Ein Eclipse-Plugin für verteilte Paarprogrammierung
  - Funktionalitäten
  - Architektur
  - Sitzungen
  - Awareness
  - Implementierung
  - Evaluation

Paarprogrammierung (PP)

- *„Pair programming - two programmers working side-by-side at one computer collaborating on the same design, algorithm, code or test...“* [2]



- Zwei Entwickler arbeiten an einem Artefakt
  - *Driver* bearbeitet das Artefakt aktiv mit Maus und Tastatur bzw. Stift.
  - *Observer* schaut zu, kontrolliert Eingaben und gibt Rat.
  - Artefakt ist meistens Code
  - Rollen wechseln sich ab

- Vor allem bekannt als Teil von Extreme Programming (XP). [9]

Warum Paarprogrammierung?

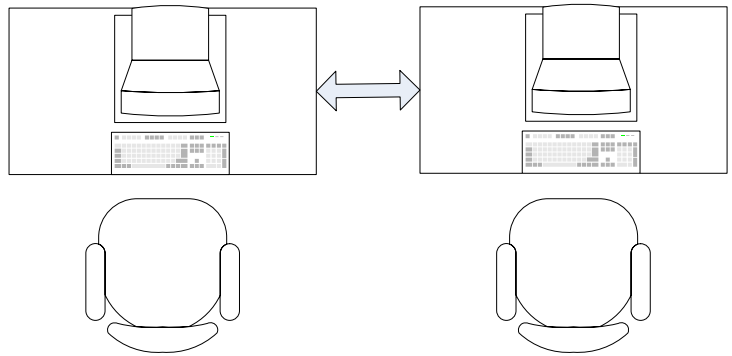
Freie Universität Berlin

- Vorteile
  1. Weniger Defekte (Stetige Durchsicht) [2, 3]
  2. Besserer Entwurf [4, 5]
  3. Höhere Motivation [2, 3, 6]
  4. Wissensaustausch [2, 7]

- Nachteile
  1. „Weniger effizient als zwei Einzelprogrammierer“ [2, 3, 6, 8]
  2. Erfordert räumliche Nähe [14]
  3. Erfordert passende Arbeitsplätze [9]

Verteilte Paarprogrammierung (DPP)

- *„… they have to see the same part of code not on one shared screen but on their individual screens. At least one of them has to be able to change this code…“* [10]



1. Warum sollte man DPP anwenden?

- Welche Stärken des PP lassen sich übernehmen?
- Welche Schwächen des PP lassen sich abmindern?

2. Wie kann man DPP anwenden?

- Werkzeugansätze der *Computer Supported Cooperative Work* [11, 12]
  - Desktop Sharing
  - Collaboration Awareness

Warum verteilte Paarprogrammierung?

Freie Universität Berlin

- Vorteile
  - Unterstützung virtueller Teams [13, 14, 15]
  - Bequemere Arbeitsplätze [7]
  - Parallele Arbeit des Paars [7]
    - 3 Ebenen unterscheidbar [*]
      - Programmebene
        (Observer darf andere Programme benutzen)
      - Sichtebene
        (Observer darf eigenständig im Code stöbern)
      - Schreibebene
        (Gleichzeitiges Schreiben)
- Nachteile
  - Weniger *Awareness* bzw. Kommunikation [11]
  - Werkzeugeinsatz [*]

Technische Ansätze

- Desktop Sharing
  - Arbeitsfläche eines Benutzers wird auf den Computern anderer Benutzer gespiegelt
  - Vorteile (im Vergleich zur Collaboration Awareness)
    - Werkzeuge auf Arbeitsfläche müssen nicht angepasst werden [11, 16]
    - Für Demonstrationen geeignet [11]
- Collaboration Awareness
  - Kollaboration direkt in Werkzeug integriert
    - Kollaborative Texteditoren [*]
    - IDEs mit DPP-Unterstützung [*]
  - Vorteile (im Vergleich zum Desktop Sharing)
    - Spezielle PP-Funktionalitäten [11]
    - Keine Probleme mit Auflösungen bzw. Bildwiederholrate [7, 12, 15]
    - Höhere Parallelität der Paararbeit [*]

Online-Umfrage

Freie Universität Berlin

- Forschungsfrage

  „In welcher Weise setzen Benutzer verteilte Paarprogrammierung heute ein und welche Funktionen eines DPP-Werkzeugs werden dabei als besonders wichtig erachtet?“

- Ankündigung über Communities zu den Themen Extreme Programming, Paarprogrammierung und DPP-Werkzeuge

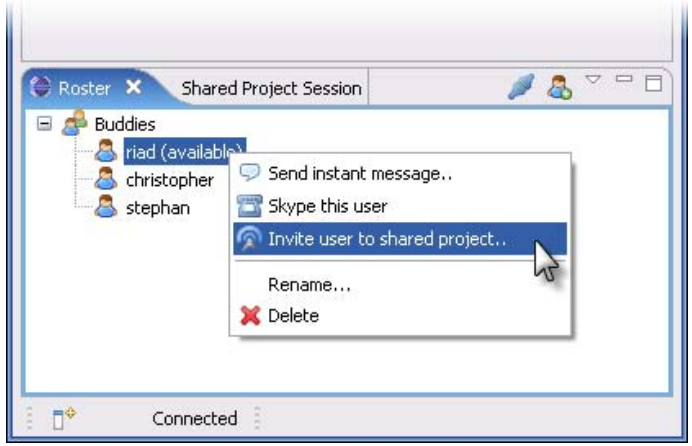
- 44 Teilnehmer
  - Die meisten programmierten Java (23%)
  - Durchschnittlich 12 Jahre Berufserfahrung
  - Meistens Bachelor- (40%) bzw. Master-Abschluss (25%)

Umfrage – Einige Ergebnisse zu Priorisierung von Anforderungen

Freie Universität Berlin

- Kommunikation
  - Sehr wichtig: Chat und Stimmübertragung
  - Wichtig: Whiteboard
  - Weniger wichtig: Webcam
- Parallele Paararbeit
  - Wichtig: Eigenständiger Aufmerksamkeitsfokus (Parallelität auf Sichtebene)
  - Weniger wichtig: Gleichzeitiges Schreiben (Parallelität auf Schreibebene)
- Awareness
  - Sehr wichtig: Observer weiß über Driver Bescheid
  - Weniger wichtig: Driver weiß über Observer Bescheid

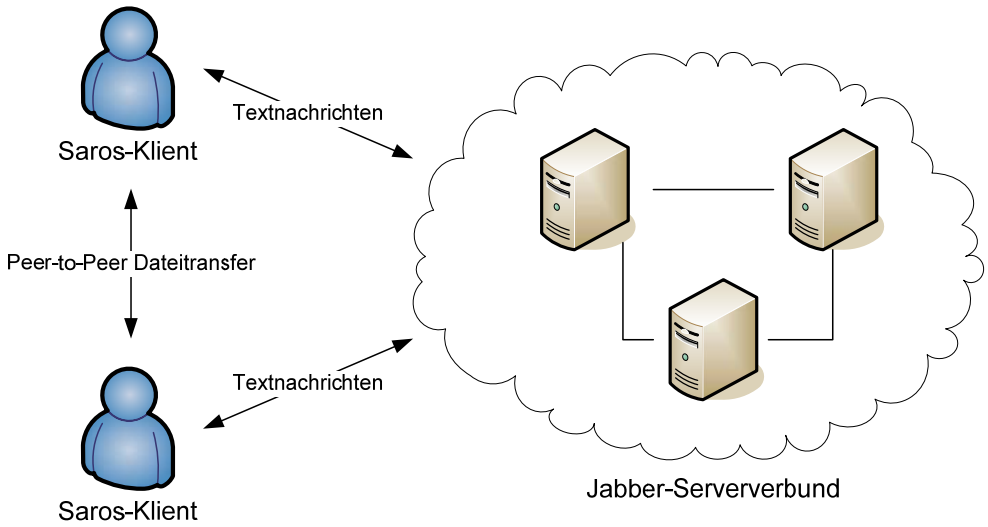
- Funktionalitäten
  - Kontaktliste mit Instant Messaging
  - Kollaborative Textbearbeitung
  - Kollaborative Dateioperationen
  - Übersetzbarkeit
  - Unterstützung für einfache Paare (ein Driver, ein Observer)
  - Sichtparallelität, aber keine Schreibparallelität
    - Verfolger-Modus
  - Protokollierung durch Erweiterung des Eclipse-ElectroCodeoGram-Sensors um neue Mikroprozesse [17]



➢ ~8000 LOC (ohne Kommentare und leere Zeilen), ~100 Klassen

Saros - Architektur

- Basiert auf Extensible Messaging and Presence Protocol (XMPP), bekannt durch Jabber.
- Aufbau
  - Server-basierte Architektur
  - XML-Nachrichten
  - Peer-to-Peer Dateiübertragungen
- Gründe
  - Offener Standard
  - Ausgereift und etabliert
  - Leicht erweiterbar



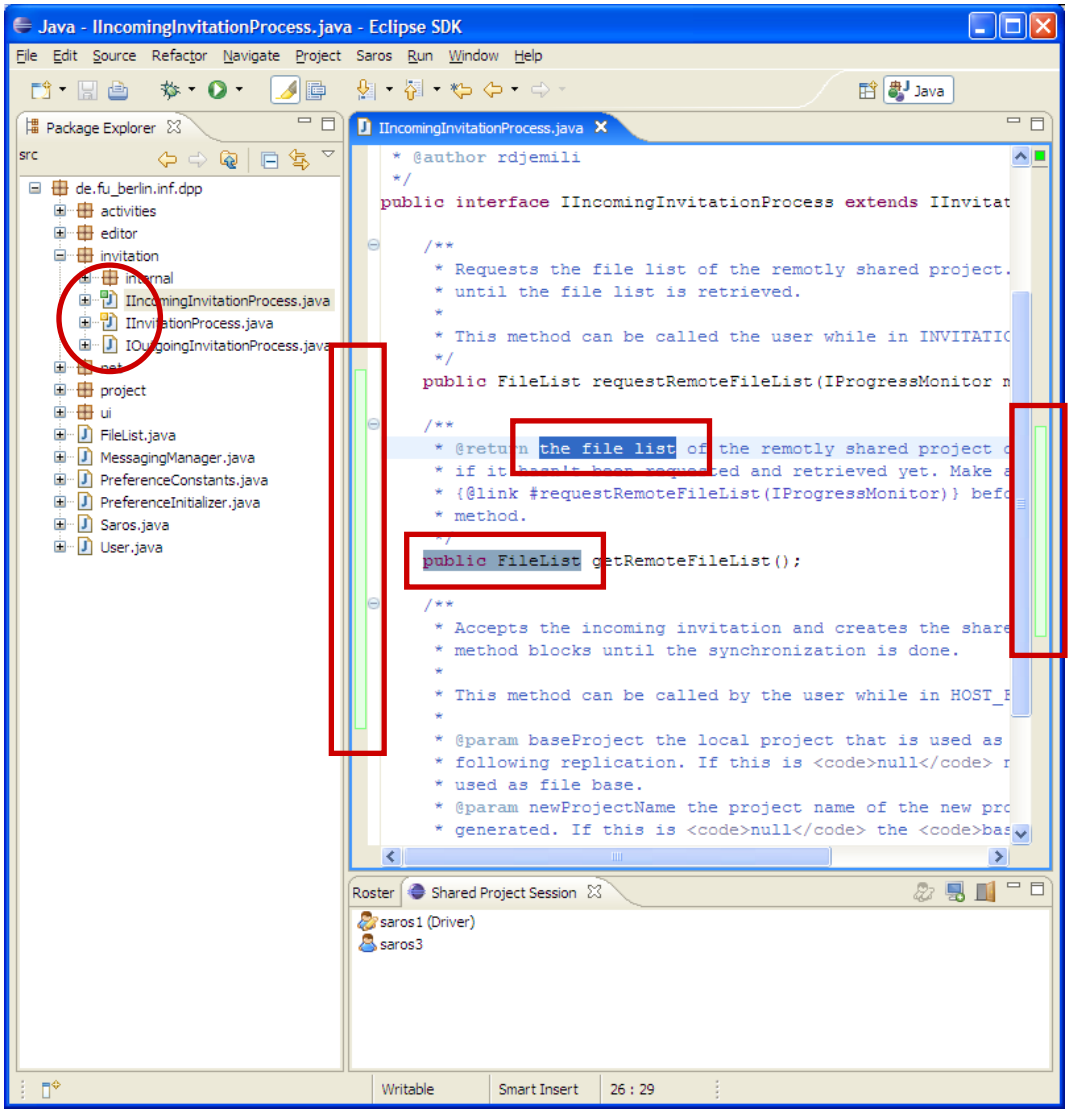
Saros – Wie funktionieren Sitzungen?

Freie Universität Berlin

- Übersetzbarkeit basiert auf Replikation [*]
  - Vorteile (im Vergleich zu virtuellen Zugriffen)
    - Übersetzer muss nicht angepasst werden
    - Keine Verzögerungen während der Arbeit
    - Programme ausführbar
  - Nachteil
    - Längere Einleitungsphase(!?)

- Wichtigste Phasen der Einladung
  1. Einlader verschickt Einladung mit eigener Dateiliste
  2. Empfänger vergleicht Dateiliste mit allen lokalen Projekten und wählt (automatisch) Projekt mit größter Übereinstimmung
  3. Das lokale Projekt wird dupliziert
  4. Dateisynchronisation zwischen den beiden Projekten.

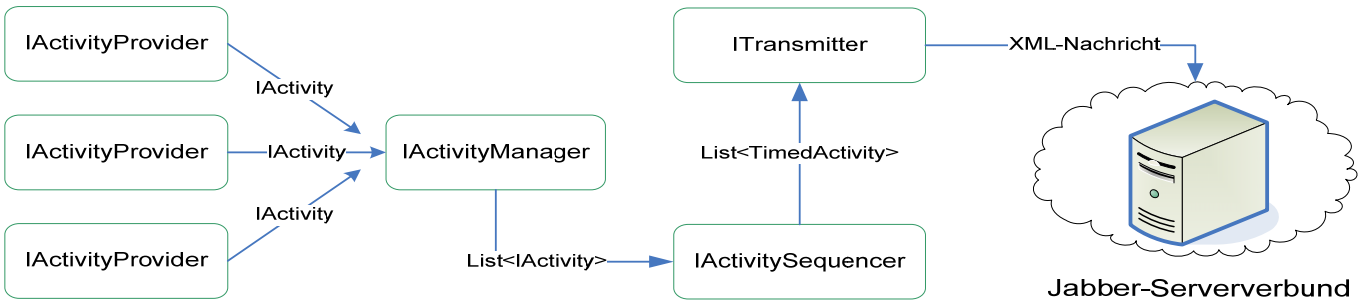
Saros – Awareness (Driver → Observer)



1. Geöffnete/ fokusierte Datei(en)
2. Cursorposition/ Textauswahl
3. Änderungen werden farblich hinterlegt
4. Sichtbereich

Saros – Aspekte der Implementierung

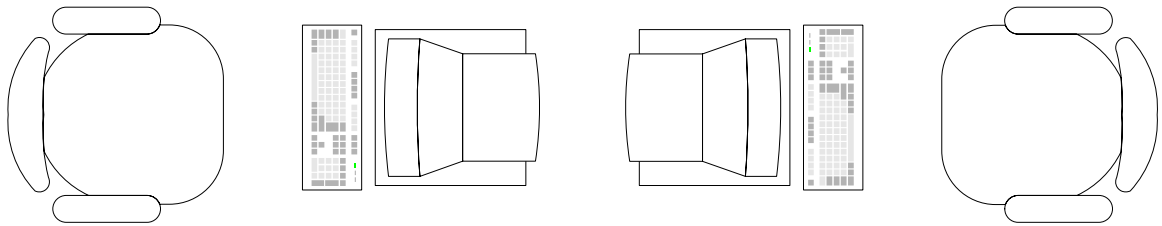
- Problem der Divergenz → Nutzung von Zeitstempeln
  - XMPP garantiert keine Reihenfolge
  - Dateiübertragungen laufen länger als andere Vorgänge
- Problem komplexer APIs
  - Nutzung von übersichtlichen Fassaden-Schnittstellen
- Erweiterbarkeit
  - Vorgänge werden als Klassen (IActivity) realisiert
  - Vorgänge werden durch Vorgangserzeuger erzeugt und verarbeitet (IActivityProvider)
  - Neue Vorgangserzeuger können einfach (auch durch neue Plugins) eingebracht werden



Erzeugen und Senden von Vorgängen

Evaluation

- Akzeptanztest, Eigene Tests und Testsitzung
- Programmierung einer Testaufgabe
  - 45 Minuten
  - Räumliche Nähe (Laut Umfrage: 33% benutzen DPP bei räumlicher Nähe, 40% bei räumlicher Entfernung)



- Interessante Beobachtungen
  - Paralleles Schreiben nur zu Anfang vermisst
  - Textauswahl wurde als Gestikersatz benutzt
    - Ähnliche Funktionalität auch für Observer sinnvoll!
  - Viel Stimmkommunikation benutzt

Zusammenfassung/Ausblick

Freie Universität Berlin

- Zusammenfassung
  - Paarprogrammierung
    - Vor-/Nachteile aufbereitet
  - Verteilte Paarprogrammierung
    - Motiviert durch Vergleich mit Vor-/Nachteilen der PP
    - Implementierungsansätze identifiziert
  - Umfrage
    - Annahmen kontrolliert/Anforderungen priorisiert
  - Eclipse-Plugin
    - Anhand erarbeiteter Anforderungen implementiert
- Ausblick
  - Hinweisfunktionalität für Observer
  - Mehr Teilnehmer in Sitzung
  - Schreibparallelität
  - VoIP-Funktionalität direkt integrieren

Videos

Fragen?

Weiterführende URLs

Freie Universität Berlin

- Diplomarbeit
  - http://projects.mi.fu-berlin.de/w/bin/view/SE/ThesisDPPI
- Saros-Plugin
  - http://sourceforge.net/projects/dpp (Sourceforge-Seite)
  - http://sourceforge.net/projects/dpp/update (Update-Site)
- Mikroprozesse/Actual Process
  - http://projects.mi.fu-berlin.de/w/bin/view/SE/MicroprocessHome

Quellen 1/2

- [*] Djemili: **Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung verteilter Paarprogrammierung**. Diplomarbeit, FU Berlin, Institut für Informatik, 2006.
- [2] Williams, Laurie, Robert R. Kessler, Ward Cunningham und Ron Jeffries: **Strengthening the Case for Pair Programming**. IEEE Software, 17(4):19–25, /2000.
- [3] Nosek, John T.: **The case for collaborative programming**. Commun. ACM, 1998.
- [4] Constantine: **L. L. Constantine on Peopleware**. Yourdon Press, Cliffs, NJ, 1995.
- [5] Williams, Laurie A. und Robert R. Kessler: **All I really need to know about pair programming I learned in kindergarten**. Commun. ACM, 43(5):108–114, 2000.
- [6] Cockburn, Alistair und Williams: **The Costs and Benefits of Pair Programming**, 2000.
- [7] Stotts, David, Williams, Nagappan, Baheti, Jen, Jackson: Virtual Teaming: **Experiments and Experiences with Distributed Pair Programming**. XP/Agile Universe 2003
- [8] Padberg, Frank und Matthias M. Müller: **Analyzing the Cost and Benefit of Pair Programming**. In: METRICS '03: Proceedings of the 9th International Symposium on Software Metrics, Seite 166, Washington, DC, USA, 2003. IEEE Computer Society.
- [9] Beck: **Extreme Programming Explained: Embrace Change**. Addison-Wesley, 1999.
- [10] Schümmer und Schümmer: **Support for Distributed Teams in eXtreme Programming**. In: eXtreme Programming Examined. Addison Wesley, 2001.
- [11] Hanks: **Empirical Studies of Distributed Pair Programming**. Doktorarbeit, University of California, Santa Cruz, 2005.
- [12] Ho, Chih-Wei, Somik Raha, Gehringer und Williams: **Sangam: a distributed pair programming plug-in for Eclipse**. In: eclipse '04: Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange, New York, NY, USA, 2004. ACM Press.
- [13] McMahon, Paul E.: **Virtual Project Management: Software Solutions for Today and the Future**. Boca Raton: St. Lucie Press, An Imprint of CRC Press LLC, 2001.
- [14] Maurer: **Supporting Distributed Extreme Programming**. In: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002, London, UK, 2002. Springer-Verlag.

Quellen 2/2

- [14] Maurer, Frank: **Supporting Distributed Extreme Programming**. In: Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002, Seiten 13–22, London, UK, 2002. Springer-Verlag.
- [15] Maurer, F. und S. Martel: **Process Support for Distributed Extreme Programming Teams**, 2002.
- [16] Greenberg, S.: **Sharing views and interactions with single-user applications**. In: Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems, Seiten 227–237, New York, NY, USA, 1990. ACM Press.
- [17] Schlesinger, Frank: **Protokollierung von Programmiertätigkeiten in der Eclipse-Umgebung**. Diplomarbeit, Freie Universität Berlin, Institut für Informatik, November 2005.

Vielen Dank!