Freie Universität Berlin

Weiterentwicklung eines Werkzeuges zur verteilten, kollaborativen Softwareentwicklung

Christoph Jacob

Institut für Informatik

FU Berlin

04.06.09

Agenda

- Grundlagen
- Saros
  - Was ist Saros?
  - Entstehungsgeschichte von Saros
  - Verwendete Technologien
- Diplomarbeit DPPIV
  - Durchgeführte Tätigkeiten
  - Kategorisierung der gefundenen Defekte
  - Empirische Bewertung
- Offene Probleme / laufende X-Arbeiten im Projekt DPP

Grundlagen

- Paarprogrammierung
  - An einem Computer
  - Driver
  - Observer
- Verteilte Paarprogrammierung
  - Räumliche Distanz (virtuelle Teams)
  - Werkzeuge ermöglichen ein gleichzeitiges Arbeiten an den gleichen Artefakten
- Verschiedene Ansätze bei verteilter Paarprogrammierung
  - Desktop Sharing
  - Collaboration Awareness
- Ebenen der Parallelität
  - Programmebene
  - Sichtebene
  - Schreibebene

Was ist Saros?

- Werkzeug für verteilte, kollaborative Softwareentwicklung
  - Collaboration Awareness, Eclipse Plugin
  - Replikationsansatz -> Quellcode kann bei allen Entwicklern übersetzt und ausgeführt werden
  - Gleichzeitiges Schreiben möglich (Parallelität auf Schreibebene)
  - Theoretisch beliebig viele Driver und Observer möglich
  - Den Enwicklern kann während einer Programmiersitzung Schreibrechte erteilt und entzogen werden (nur Host)
  - Theoretisch nicht auf bestimmte Programmiersprachen beschränkt (dennoch im Moment nur Unterstützung von JDT und CDT)
- keine Integration von Audio und/oder Videoübertragung
  - -> hierfür Skype, Mumble oder ähnliche SW verwenden

Entstehungsgeschichte von Saros

Freie Universität Berlin

- Diplomarbeit von R. Djemili *(DPPI)*
  - Titel: „Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung verteilter Paarprogrammierung“
  - Ein Driver
  - Ein Observer
- Studienarbeit von B. Gustavs *(DPPII)*
  - Titel: „Weiterentwicklung eines Eclipse-Plugins zur verteilten Paarprogrammierung“
  - Ein Driver
  - Mehrere Observer
- Diplomarbeit von O.Rieger *(DPPIII)*
  - Titel: „Weiterentwicklung [...] im Hinblick auf Kollaboration und Kommunikation“
  - Mehrere Driver (Parallelität auf Schreibebene)

Verwendete Technologien

Freie Universität Berlin

- Eclipse als IDE
  - Mächtige IDE, weit verbreitet
  - Erweiterbarkeit durch Plugin-Architektur
- XMPP als Nachrichtenprotokoll / Jabber-Serververbund
  - Erweiterbarer etablierter Standard auf XML Basis
  - Dezentraler Serververbund (Jabber)
  - Smack als Java-Bibliothek
- Jupiter-Algorithmus als Nebenläufigkeitskontrolle
  - Operational Transformation (OT) Verfahren
    -> keine Sperren, lokale Veränderungen sofort angewendet
  - Sichert Konvergenz und Kausalitätserhaltung
  - Anders als bei anderen OT-Verfahren besitzt Jupiter eine sternförmige Topologie, d.h. zentraler Server mit dem die Teilnehmer (Clients) kommunizieren

Durchgeführte Tätigkeiten im Rahmen der Diplomarbeit DPPIV

Freie Universität Berlin

- Erneute Anforderungsbestimmung und -analyse
  - Betreuer nahmen die Rolle des Kunden ein
- Bestimmung von Versagen der Software
  - 2 durchgeführte Testexperimente mit externen Entwicklern als Tester
  - Funktionstests (Black-Box) anhand von Anwendungsfällen
  - Strukturtests (White-Box)
  - Codedurchsichten
- Ausbesserung der dazugehörigen Defekte
- Erweiterung der Software um eine erweiterte Konsistenzsicherung
- Neuimplementierung der P2P-Verbindungen mit UDP/RUDP als Transportprotokoll
- Empirische Bewertung der Software durch eine Mini-Fallstudie (siehe übernächste Folie)

Kategorisierung der Defekte

- Ungeeignete Datenstrukturen
- Inkorrekte oder fehlende Synchronisation
- Falsche Verwendung von asynchronen Methoden
- Zu häufige Verwendung von Interfaces
- Altlasten im Quelltext

Mini-Fallstudie (Ziel und Fragen)

Freie Universität Berlin

- *Ziel:* erste Erkenntnisse für eine Validierung der Software
- *Fragen:*
  1. Wird eine hinreichende Interaktion zwischen den Entwicklern für ein gemeinsames Arbeiten ermöglicht?
  2. Möglichkeit des gleichzeitigen Schreibens von Entwicklern positiv bewertet?
  3. Für welche Entwicklungstätigkeiten wird Saros als nützlich, für welche als weniger nützlich empfunden?
- Ausgehend von diesen Fragen wurde ein empirisches Experiment entworfen

Mini-Fallstudie (Durchführung)

- *Teilnehmer:*
  - 10 Studenten des Instituts
  - 2 Teams mit n=3 und 2 Teams mit n=2
- *Szenario:*
  - unterschiedlichen Räume
  - Skype für eine Audiokonferenz
- 3 Programmieraufgaben
  1. Sehr leichte Aufgabe um Werkzeug kennen zu lernen
  2. Programmieraufgabe aus ACM Programming Contest
  3. gemeinsame Codedurchsucht um Defekte zu finden
- Datenquellen der Qualitativen Untersuchung
  - Fragebögen vor und nach der Programmiersitzung
  - Befragungen in einer anschließenden Diskussionsrunde
  - Meine Beobachtungen während des Experiments

Mini-Fallstudie (Ergebnisse I)

- *Ergebnisse:*
  1. *Interaktionsmöglichkeiten wurden von allen als ausreichend für ein gemeinsames Arbeiten am Quelltext empfunden*
     - *Videoübertragung* wurde sich nur von sehr wenigen Teilnehmern gewünscht und wenn dann nur um sein Gegenüber ein wenig kennen zu lernen
     - Eine *Skizzenfunktion*, zum gemeinsamen erarbeiten von Skizzen oder zumindest die Möglichkeit Skizzen zu übertragen wurde von fast allen vermisst, ohne dass da nach explizit gefragt worden ist
     - *Textnachrichten* wurden gar nicht verwendet, die Kommunikation fand ausschließlich über Audio statt. Nur als Fallback wichtig, falls Audio nicht funktioniert

Mini-Fallstudie (Ergebnisse II)

2. *Mehrschreiberfunktionalität wurde sehr unterschiedlich verwendet und bewertet*
   - Einige Gruppen haben sehr stark davon Gebrauch gemacht, andere dagegen sehr wenig
   - Vermutung, dass die Erfahrung in PP/DPP sowie im Umgang mit Saros einen nicht unerheblichen Einfluss darauf hat
3. *Das Werkzeug wurde für gemeinsame Programmiersitzungen sowie für Schulungen als nützlich angesehen*
   - Ein Teilnehmer gab an, dass Saros die Möglichkeit schafft, für einige Zeit gemeinsam an Code-Abschnitten zu arbeiten und dann wiederum für einige Zeit unterschiedliche Stellen zu bearbeiten
   - Interaktiver Unterricht sei möglich (kein Frontal-Unterricht)
   - Bei der Verwendung für Code-Durchsichten gingen Meinungen auseinander, einige gaben eine zu unterschiedliche Lesegeschwindigkeit als Begründung dafür an, dass Saros hierfür nicht geeignet sei

Offene Probleme / Aufgaben

- Immer noch Probleme mit NAT-Traversierung bei Direktverbindungen
  - Jingle-Implementierung der Smack-Bibliothek scheint fehlerbehaftet zu sein
  - -> laufende Diplomarbeit von Sandor Szücs *(DPPVI)*
- Keine brauchbare Undo/Redo Funktionalität
  - Sollte nur eigene Veränderungen berücksichtigen
  - -> laufende Diplomarbeit von Sebastian Ziller *(DPPVII)*
- Benutzerfeedback in Saros integrieren
  - Umfragen und Aufzeichnung von Mikroprozesse
  - -> laufende Bachelorarbeit von Lisa Dohrmann *(DPPVIII)*

Vielen Dank!