Konzeptvorstellung DPP III

Oliver Rieger
Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/w/SE/ThesisDPPIII

- Anforderungen
- Existierende Systeme
- Konzept-Ansätze
- Evaluierung
- Zeitplan
- Vorläufige Gliederung

Zweck des Projekts DPP III

- Software-Entwicklung an verteilten Standorten
- Vorteile der Paarprogramming auch dieser Situation nutzen zu können
- "Klassische Vorgaben" des Pairprogramming dernneuen Einsatz-Szenarien anpassen
  - Direkte Kommunikation ist Hauptbestandteil
  - im verteilten Umfeld durch Nutzung vielfältiger Kommunikationsmittel „Nachteile" für verteilte Paarrprogramming kompensieren
  - strikte Rollenverteilung von Driver und Observer „auflockern"
  - Erkenntnis-Gewinnung über verteilte Paarprogrammierung durch ECG-Informationen

Ziele des Projekts DPP III

- Infrastruktur für kollaborative Schreibzugriffe
  - gleichzeitiger Schreibzugriff unter Einhaltung der Konsistenzanforderungen aller relevanter Projektdateien
- Ausbau der Kommunikationsmittel
  - Verbesserung der Kommunikation und Verstärkung des Gruppengefühls
  - Audio / Video Kommunikation ermöglichen
  - Multi-User-Chat ermöglichen
- Weiterentwicklung der bisherigen Umsetzungen
  - bisherige Umsetzung zur Realisierung der Ziele und Anforderungen verwenden und weiterentwickeln
- Whiteboard Funktionalität
  - Unterstützung in Design- und Entwurfsphasen als zusätzliches Kommunikationsmittel (niedrige Priorität ?)

Funktionale Anforderungen

- Das Projekt muss die bisherige Umsetzung der Paarprogrammierung sinnvoll ergänzen.
- Das Projekt muss einen kollaborativen Schreibzugriff von mindestens zwei Drivern ermöglichen.
- Die Veränderungen der einzelnen Driver müssen eindeutig zugeordnet werden können.
- Das Projekt muss eine erweiterte Kommunikation mit Audio und Video ermöglichen.
- Das Projekt muss eine konfigurierbare eindeutige Farbzuweisung aller Anwender gewährleisten.
- Das Projekt muss eine Whiteboard Funktionalität beinhalten, um kreative Prozesse in der Paarprogrammierung besser zu unterstützen.

Funktionale Anforderungen (2)

- Das Projekt muss Aktionen eines beobachteten Drivers im Package-View des veröffentlichten Projekts kommunizieren.
- Das Projekt muss einen Dienst für die Nebenläufigkeitskontrolle beinhalten, der während der Projekt-Initialisierung automatisch gestartet wird.
- Das Projekt muss zu jedem Zeitpunkt visualisieren, welcher Teilnehmer zur Zeit welche Projektdatei bearbeitet.
- Das Projekt muss eine Historie Funktion der durchgeführten Änderung mit zugehöriger Autoren-Information bereitstellen.
- Das Projekt muss der Rolle Observer die Wahl des zu beobachtenden Driver ermöglichen.

Nichtfunktionale Anforderungen (1)

- Das Projekt muss auf die Projekte DPP I und DPP II aufbauen und die bestehende Umsetzung verwenden und erweitern.
- Die nichtfunktionalen Anforderungen aus DPP I müssen auch in DPP III erfüllt werden.
- Die Konsistenz der Projektdateien muss immer gewährleistet sein, auch nach Verbindungsabbrüchen müssen entstandene Konflikte aufgelöst werden können.
- Die Kommunikation muss auch in komplexen Netzwerk-Topologien sichergestellt sein.
- Die Kommunikation im Projekt muss mit einer für den Anwendern akzeptablen Latenzzeit sichergestellt werden.

Anwendungsfall-Modell (1)

Akteure im Anwendungsfall-Modell

- **Projekt-Host**
  - erstellt die Projekt-Sitzung und läd Teilnehmer aus seiner Kontaktliste für diese Sitzung als Teilnehmer ein.
  - Projekt-Host ist automatisch **Driver** mit Schreibrechten auf alle Projektdateien
- **Driver**
  - aktiver Teilnehmer an der Projekt-Sitzung und kann auf alle Projektdateien schreibend zugreifen.
- **Observer**
  - ein passiver Teilnehmer an der Projekt-Sitzung
  - besitzt keine Schreibrechte auf Projektdateien
  - Verfolgermodus für Driver

Anwendungsfall Modell (2)

- Der **Projekt-Host** veröffentlicht ein Projekt zum Start der Sitzung. Im folgenden lädt er **Teilnehmer** für diese Sitzung ein, die bei Annahme zunächst als **Observer** der Sitzung hinzugefügt werden. Während dieses Vorgangs wird das Projekt lokal beim **Teilnehmer** aktualisiert. Somit haben zu Beginn der Sitzung alle Teilnehmer eine konsistente Projektversion.
- Ein **Teilnehmer** der Sitzung wechselt seine Rolle von **Observer** zu **Driver** und wird als zweiter **Driver** mit Schreibrechten auf alle Projektdateien in der Sitzung registriert. Beiden aktuellen Driver können unabhängig von einander Projektdateien bearbeiten, wobei alle Änderungen sofort an alle Teilnehmer der Sitzung kommuniziert werden. Alle Teilnehmer mit der Rolle **Observer** können im Verfolger-Modus einen beliebigen Driver verfolgen.
- Ein **Driver** wechselt in einen Driver-Verfolger-Modus und kann somit wie ein Observer dem anderen **Driver** folgen. Er hat jedoch die Möglichkeit jederzeit schreibend auf die geöffneten Dateien zuzugreifen.

Anwendungsfall Modell (3)

- Beide **Driver** bearbeiten zeitgleich die gleiche Datei, wobei es zu konkurrierenden Schreibzugriffen während der Bearbeitung kommt. Die lokalen Operationen jedes **Drivers** werden an den **Projekt-Host** übertragen und Konflikte werden automatisch aufgelöst. Notwendige Operationen zur Konsistenzerhaltung werden vom **Projekt-Host** automatisch an alle **Teilnehmer** übertragen und dort lokal ausgeführt. Ein manuelles Eingreifen seitens der Teilnehmer ist nicht notwendig. Textuelle Veränderungen der Driver werden in der entsprechenden eindeutigen Farbmarkierung hervorgehoben.
- **Driver** können zur Unterstützung weitere Kommunikationsmittel über die Oberfläche starten. Neben der (Multi-) Chat-Kommunikation können die Driver untereinander mit Audio- und Video-Kommunikation direkter und effizienter Zusammenarbeiten.

Funktionen der aktuellen Umsetzung

- Gemeinsames Arbeiten eines Drivers mit mehreren Observern an einem gemeinsamen Projekt
- Kommunikation basierend auf XMPP Datenaustausch
- Projekt-Abgleich beim Hinzufügen neuer Teilnehmer
- Projekt kann jederzeit lokal ausgeführt werden
- Farbige Markierung von Text-Selektionen, Sichtbereich und Änderungen
- Chat-Funktion als zusätzliches Kommunikationsmittel
- Rollenwechsel zwischen Driver und Observer
- Verfolger-Modus für Observer
- Sitzungsinformationen von Teilnehmern und der entsprechenden Rollen

DEMO

Einschränkungen der aktuellen Umsetzung

- Kein kollaborativer Schreibzugriff
  - Wechsel des exklusiven Schreibzugriffs durch „take driver role“
- Verwendung einer älteren XMPP API (Smack 2.2.1)
  - in komplexeren Netzwerk-Topologien kein direkter File Transfer möglich
  - Weiterentwicklung im Bereich Audio/Video nicht nutzbar
  - automatisches Reconnect nicht nutzbar
- Initialisierung größerer Projekte
  - bei Initialisierung größerer Projekte wird IDE blockiert
  - fehlende Status Informationen über den Verlauf

Existierende CSCW-Systeme (1)

- **SubEthaEdit**
  - ein von Apple entwickelter kommerzieller kollaborativer Gruppen-Editor mit vielfältigen Awareness-Informationen
  - verwendet eine peer-to-peer Kommunikation auf Grundlage des BEEP Protokolls und Bonjour zum automatischen Auffinden veröffentlichter Dokumente
  - gleichzeitiger Schreibzugriff mit optimistischer Zugriffskontrolle
- **ACE (a collaborative editor)**
  - verwendet wie SubEthaEdit Bonjour und BEEP für eine peer-to-peer Kommunikation
  - blockierungsfreier Schreibzugriff mit optimistischer Zugriffskontrolle
  - Operational Transformation Realisierung durch Jupiter Algorithmus

Existierende CSCW-Systeme (2)

- **IRIS**
  - von der TU München entwickelter kollaborativer Gruppen-Editor
  - in der Neuentwicklung auf Java-basierendes CSCS-System mit optimistischer Zugriffskontrolle
  - beinhaltet keine Algorithmus zur automatischen Konfliktauflösung
  - Dokumenten werden in einer baumartigen Dokumentenstruktur aufgebaut
  - beinhaltet Historie-Informationen zur Rekonstruktion des Änderungsverlaufs

Existierende CSCW-Systeme (3)

- **Eclipse Communication Framework (ECF)**
  - Kommunikationplattform auf Basis eines Kommunikations-Servers unter Verwendung des **Equinox** Projekts, um OSGi (Open Services Gateway Initiative) auf dem Server zu realisieren
  - Abstraktion der Kommunikationsprotokolle als ECF-Protokol-Adapter
  - Bsp: FileTransfer, Remote (OSGi) Service, Chat, Presence, Jingle, VoIP, zero-conf discovery
- **Eclipse Shared Text Editor (collab.editor)**
  - Umsetzung eines kollaborativen Editor ohne Nebenläufigkeitskontrolle mit ECF
- **Real-Time Shared Editor**
  - basierend auf collab.editor mit Nebenläufigskontrolle

Konzept-Ansätze (Nebenläufigkeit)

- optimistische Zugriffskontrolle ohne Sperr-Mechanismen
  - lokale Veränderungen werden ohne Verzögerung oder Blockierung ausgeführt
  - entfernte Operationen werden zur Konsistenz-Sicherung umgewandelt
  - jeder Autor hat zu jeden Zeitpunkt immer denselben Stand des Dokuments
- Operational Transformation zur Konsistenz-Sicherung
  - automatische Konsistenz-Erhaltung mit OT
  - Jupiter collaborative editing system, ein stark verbreiteter Synchronisationsalgorithmus
  - verwendet veränderte Version des dOPT (distributed Operational Transformation) Algorithmus zur Verwendung in Client-Server-Umgebungen
  - 2-way-Prinzip verringert Kommunikationsaufwand

Konzept-Ansätze (Kommunikation)

- Client-Server-Ansatz für Synchronisation
  - geringerer Kommunikationsaufwand
  - geringere Komplexität (siehe hierzu Jupiter)
  - Projekt-Host als Server-Instanz
- Ausbau der Kommunikation auf Basis von Smack 3.x
  - Verwendung notwendiger neuer Erweiterungen (Bsp: Jingle, FileTransfer, ...)
  - Realisierung der Server-Komponente durch eingebettet Logik im bestehenden Plugin

ODER

- Realisierung auf Basis des ECF Servers
  - Integration der bestehenden XMPP Kommunikation
  - Verwendung existierender Protokoll-Adapter und Projekte

Konzept-Ansätze (Synchronisation)

- Paarprogrammierung an gemeinsamen Dateien
  - bisheriger **synchroner** Ansatz wird weiterverfolgt
  - alle Änderungen werden sofort an alle Teilnehmer kommuniziert
- Viele Driver arbeiten an unterschiedlichen Dateien
  - in diesem Szenario kann es zu einem sehr großen Kommunikationsaufwand kommen
  - verfolgen eines **asynchronen** Ansatzes
  - nur beobachtete Dateien werden sofort synchronisiert
  - Synchronisation aller Dateien vor Ausführung des Kompilier-Prozesses
- Historien-Informationen
  - zum asynchronen Abgleich veränderter Dateien Verwendung einer Historie mit User-ID und Zeitpunkt

Offene Fragen (1)

- Schreibzugriff für zwei Teilnehmer?
  - eng am Konzept der Paarprogrammierung
  - synchroner Kommunikationsabgleich aller Dateien zu jedem Zeitpunkt

  ODER
  - bei mehr als zwei Teilnehmer, die unterschiedliche Dateien bearbeiten steigt Kommunikationsaufwand stark an
  - Verwendung einer teilweise asynchronen Kommunikation mit Historie Daten?
- Verfolger-Modus für Driver?
  - synchronisiertes Öffnen der Dateien bei gleichzeitigen Schreibrechten

Offene Fragen (2)

- Priorität einer gesicherten Verbindung?
  - durch eine Vielzahl von Kommunikationsmittel erschwerte Umsetzung
  - bei Open Source Projekten Quellcode offen, Verschlüsselung trotzdem sinnvoll?

Evaluierung

- Kurze Evaluierungs-Runden Projekt-begleitend in den regelmäßig stattfindenden Projekt-Sitzungen
- Nach Abschluss der Haupt-Entwicklungsphase größere Evalierungs-Runde
  - mindestens drei Teilnehmer
  - längerer Zeitraum von mindestens zwei Stunden
  - vordefinierte Einsatzszenarien / Aufgaben, die in diesem Zeitraum bearbeitet werden
  - verteilte Standorte, die sich sowohl innerhalb des UNI-Netzwerks als auch in Weitverkehrs-Netzwerke befinden

Vorläufige Gliederung

1. Einleitung
2. Analyse des bestehenden Systems
3. Grundlagen CSCW-Systeme
    1. Allgemeine Grundlagen
    2. Existierende Umsetzungen
4. Konzeption für die Umsetzung
    1. CSCW im Kontext der Paarprogrammierung
    2. Netzwerkkommunikation
    3. Kollaboration
    4. Weitere Anforderungen
5. Entwurf und Implementierung
6. Evaluierung
7. Literaturverzeichnis

Zeitplan

- Einarbeitung in DPP I + DPP II (1-2)
- Analyse bestehender Probleme und Einschränkungen (3-5)
- Grundlagen CSCW-Systeme (6-7)
- Analyse existierender CSCW-Systeme (8-9)
- Grundlagen Algorithmen (10-11)
- Konzeption / Anpassung der Kommunikationsstruktur (12-20)
- Konzeption und Implementierung der Kollaboration (12-20)
- Ausarbeitung und Durchführung der Evaluierung (21-22)
- Überarbeitung der Ausarbeitung (23-24)
- Abschlusskorrektur (25)
- Abgabe (26)

Danke!