Freie Universität Berlin

Saros - Distributed Collaborative Editing and Distributed Party Programming

Semi-lazy & Partial File-sharing in Saros (Antrittsvortrag/Konzeptvorstellung)

Masterarbeit - Arbeitsgruppe Software Engineering

Karl Held

Abgabetermin: 16.09.2011

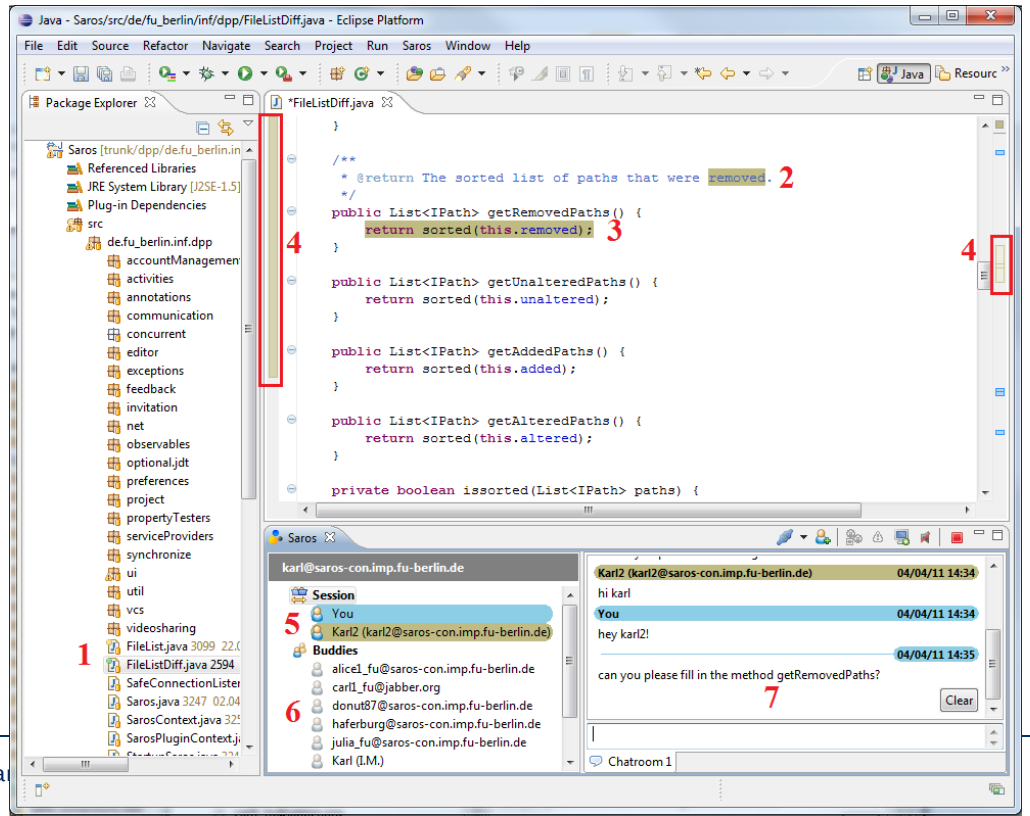
Datum: 14.04.2011

Agenda

- Kurzvorstellung Saros
- Motivation
- Aufgabenstellung
- Status quo
- Konzept
- Ablauf
- weiteres Vorgehen

Kurzvorstellung Saros

- Plugin für Eclipse Entwicklungsumgebung
- Projekte gemeinsam bearbeiten
  - » verteilte Paarprogrammierung, sowie Programmierung mit verteilten Beteiligten (> 2)
  - » verteiltes Reviewen
- Saros hält Projekte auf dem gleichen Stand und informiert Teilnehmer über Änderungen



Motivation

- Saros:
  - »…ermöglicht verteilte Zusammenarbeit an Projekten
  - »…hält während einer Session alle Kopien der Projekte auf dem gleichen Stand
- jedoch:
  - »nur auf Projektebene
  - »wenn das gesamte Projekt auf einmal synchronisiert wird

- Problematik:
  - »gesamtes Projekt synchronisieren, selbst wenn eine Datei/Ordner in der Session von Bedeutung
  - »vollständige Synchronisation großer Projekte kann lange dauern → verzögert Sessionstart

Aufgabenstellung

At the moment, Saros sharing is project-based, which is to say the smallest unit users may synchronise is a whole project. While this has it advantages, and we wish to retain this capability, it can be rather time-consuming and unnecessary for people who wish to conduct "quick" sessions. It would therefore be advantageous if Saros had the capability to synchronise at the file level also.

There are a number of very important design decisions regarding such a capability. For example, sharing only certain files poses more problems when using a statically-typed language than a dynamically-typed one. Our own proposal is that Saros should share files in a semi-lazy fashion, sharing files in the background, but prioritising some over others.

» capability to synchronise at the file level also

» share files in a semi-lazy fashion, sharing files in the background, but prioritising some over others

Status quo

- Sharen/Synchronisieren von ganzen Projekten möglich
- mehr als ein Projekt auf einmal in Session nutzbar
- Projekt(-e) auch nach Sessionstart synchronisierbar
- Synchronitätskontrolle und Korrektur auf Projektebene

Konzept

- partielle Projektsynchronisation:
  - » Sharen/Synchronisieren von Dateien, Ordnern und ganzen Projekten ermöglichen = Ressourcen
  - » Ressourcen für mehr als ein Projekt auf einmal in Session verwalten
  - » beliebige Ressourcen auch zu laufender Session hinzufügen können
  - » Synchronitätskontrolle auf Dateiebene
  - » Hervorhebung synchronisierter zu nicht synchronisierten Ressourcen für Host

- erfordert:
  - » verschiedenste weitreichende Anpassungen und Veränderungen der Benutzerschnittstelle → „Freiheit“ der Ressourcenauswahl gewährleisten
    - » Ressourcenbaum mit Checkboxen
    - » Kontextmenu „Share with“ auch auf Datei und Ordnerebene
    - » ggf. Decorator auf Datei- und Ordnerebene wenn synchronisiert
  - » Logik komplett von Projekt auf Ressourcenebene umstellen und anpassen
  - » Synchronitätskontrolle auf partielle Projektverarbeitung modifizieren

Konzept

- „Semi-lazy“ Sharing von Projekten/Ressourcen:
  - »Priorisierung von Ressourcen im Synchronisationsvorgang
    - » vom Host geöffnete Dateien / Pakete zuerst synchronisieren, den Rest danach im Hintergrund aktualisieren
    - » Dateien sofort synchronisieren, wenn diese geöffnet werden und noch nicht in der Session synchronisiert sein sollten
  - »bei größeren Projekten schneller zum Arbeiten übergehen können wenn „wichtige“ Dateien sofort, und die „Masse“ im Hintergrund synchronisiert wird
  - »gesamten Synchronisationsvorgang transparenter für beide Seiten gestalten
    - » ggf. Fortschritt und Restdauer sichtbar machen

- erfordert:
  - »Modifikation der Dateiaustauschfunktionalität um priorisieren zu können
  - »ggf. schnelleren Dateiaustausch (Zip → Stream)
    - » Filestreamproblematik lösen
  - »sinnvolle Priorisierungsansätze auswählen und evaluieren

Fokus

- Fokus bei allen Änderungen
  - » Usability/Bedienbarkeit/Brauchbarkeit
    - » Selbsterklärend gestalten (v.a. Benutzerschnittstelle)
  - » Bewährtes/Bekanntes belassen bzw. sinnvoll erweitern
  - » Effektivität
  - » Transparent/Informativ für Benutzer gestalten
  - » Konsistenz

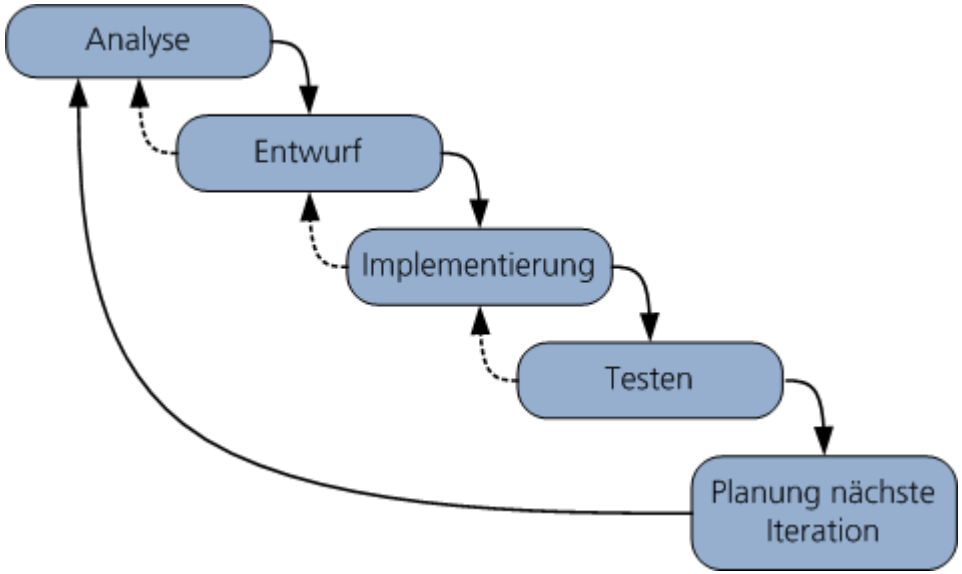
Ablauf

- Meilensteinplan

#	Name	Beschreibung	Zeitspanne in Wochen
1	Einarbeitung/Vorarbeit	Orientierung in Code und Struktur; Besprechung der Masterarbeit/Aufgabe; Formalien;	1-2
2	wissenschaftliches Vorgehen klären und Konzepte entwickeln	Meilensteinplan; UseCases; Kriterien für die Gewichtung der UseCases; Themengebiet abgrenzen; Anforderungsanalyse; Iterationen festlegen	2-3
3	Lösungsansätze entwickeln und evaluieren		4-6
4	Lösungsansätze umsetzen und Testen	empirische Bewertung vornehmen	6-8
5	Probleme ermitteln und lösen		3-4
6	Fertigstellung der Masterarbeit	textuelle Änderungen/Feinschliff	2-3

Vorgehen

- inkrementell, iterativen Vorgehensweise
- Teilaspekte lauffähig und testbar → „kleiner“ Patch für Reviewboard
- strikte Einhaltung wird nicht immer möglich sein



weiteres Vorgehen

- funktionelle Implementierung vorantreiben
- in Kommunikation mit Kollegen und Betreuern vorläufige Ergebnisse bewerten und diskutieren
- Defekte korrigieren