Studienarbeit

Weiterentwicklung des Eclipse-Plug-Ins „Saros“
zur Verteilten Paarprogrammierung
(DPPII)

Björn Gustavs
Matrikelnummer 3303120
uni@gustavs.de

Betreuer:

Christopher Oezbek
Stephan Salinger

Inhaltsverzeichnis

1. Einleitung

Saros ist ein Eclipse-Plugin zur verteilten Paarprogrammierung, welches von Riad Djemili im Rahmen seiner Diplomarbeit an der Freien Universität Berlin entwickelt wurde. Es ermöglichte das gemeinsame Arbeiten von 2 Personen an einem Projekt in Eclipse. Obwohl dies im Rahmen der Anforderungen auch umgesetzt wurde, blieben für den weiteren Einsatz noch wichtige Punkte offen und weitere Funktionalitäten wurden gewünscht.

Meine Studienarbeit soll hier eingreifen und gewünschte Funktionalitäten und Korrekturen einbringen.

In Diskussion mit den Betreuern wurden Ziele formuliert und so konnte ich mit der Einarbeitung in Saros beginnen.

2. Einarbeitung

Neben dem Studium des Quellcodes von Saros war auch die Diplomarbeit für die Einarbeitung hilfreich (Kapitel 5.3.1), da hier Klassen, Interfaces und deren Zusammenhänge beschrieben und erklärt wurden.

Ein anfängliches Treffen mit Riad Djemili war ebenfalls sehr nützlich, um so Hinweise und Tipps zum Programm selber, als auch zur Entwicklung, aus erster Hand zu erhalten. Besonders Hinweise zum Debuggen von Eclipse-Plugins waren sehr hilfreich da dies ein fester Bestandteil meiner weiteren Arbeit sein würde.

Erste tiefere Schritte mit Saros tätigte ich neben dem praktischen Einsatz von zwei Saros Instanzen auch im Debugger und verfolgte so den Programmablauf in besonderen Situationen. So kam ich mit ersten kleinen Korrekturen und Erweiterungen gut zurecht, zum Beispiel dem Entfernen der Hinweis-Messagebox bei Observern, welche einen Änderungsversuch tätigen (siehe 3.1).

Die erworbenen Einblicke in den Start der Saros-Entwicklung habe ich in die Developer-Dokumentation auf der Projekthomepage einfließen lassen, auf dass weitere Programmierer einen leichten Start haben.

3. Programmierung / Updates

In den folgenden Abschnitten sollen die Erweiterungen dokumentiert werden, welche ich für Saros vorgenommen habe.

Bei Implementierungen werde ich jeweils das Problem, die Herangehensweise sowie die Umsetzung erläutern.

3.1 Homepage

Ein Projekt benötigt eine Repräsentation im WWW. Hier fehlte Saros eine Homepage, welche über das Projekt informiert, Dokumentationen bereitstellt als auch die Installation des Plugins ermöglicht.

Ich habe im Verlauf der Studienarbeit diese Homepage als Projekthomepage auf Sourceforge erstellt und diese bietet folgende Informationen:

1. Erklärung von Saros, Nutzen und generelle Funktionsweise
2. Technische Details zur Funktionsweise
3. Screenshots zur Vorstellung von Saros
4. Dokumentation, Einstiegshilfe
5. Update Location für das Eclipse Plugin.
   Hier habe ich zum Ende meiner Tätigkeit ein aktuelles Saros-Plugin zur Installation bereit gelegt.

Die Homepage ist unter http://dpp.sourceforge.net/ zu erreichen.

3.2 Vermeiden vom Popup Dialog bei Eingabeversuch eines Observers

Erklärung:
(Bugtracker: #1541282: “Popup dialog when violating driver role is a UI sin.”)

Observer dürfen in einer Sitzung keine Änderungen an Projektdateien vornehmen. Versucht er es dennoch, meldet Saros das Fehlverhalten mittels einer Messagebox.

Plan:
Anstatt einer Messagebox soll der Hinweis auf fehlende Rechte zum Editieren weniger aggressiv in der Statusleiste angezeigt werden.

Durchführung:
Für Observer prüft ein `VerifyKeyListener` auf Tastatureingaben und gibt in diesem Falle eine Messagebox aus. Diese wurde ersetzt durch das Setzen selbiger Information in der Statusleiste.

3.3 Option für Fortführung eines existierenden Projektes bei Einladung

Erklärung:
Bei Einladung eines neuen Nutzers, wurde immer ein neues Projekt erstellt, evtl. unter Nutzung lokaler teilidentischer Projekte. Das Fortführen eines bereits lokalen Projektes als eingeladener Sitzungsteilnehmer war daher immer mit einer unnötigen Duplizierung des Projektes verbunden.

Plan:
Während der Einladung soll der eingeladene Nutzer die Option haben, das Projekt mit höchster Übereinstimmung fortzusetzen.

Durchführung:
Da in der Klasse `IncomingInvitationProcess` die Unterscheidung ein Projekt zu erstellen oder fortzusetzen bereits implementiert war, beschränkte sich diese Erweiterung auf die Anpassung der GUI des `JoinSessionWizards`.

Ich habe Radiobuttons für das Erstellen eines neuen Projektes beziehungsweise Fortsetzen eines existierenden Projektes hinzugefügt, mit einer entsprechenden Bezeichnung („Update and use existing project“) und dem zutreffenden Projektnamen beschriftet. Saros wählt zum Fortsetzen das Projekt mit der höchsten Übereinstimmung.

Wählt der Anwender das Fortsetzen eines Projektes wird der entsprechende Projektname der `accept()`-Methode des gegenwärtigen `IncomingInvitiation`-Objektes übergeben. Die weitere Unterscheidung war bereits für das Erstellen und Synchronisieren des lokalen Projektes implementiert.

3.4 Unterstützung mehrere Observer

Erklärung:
Saros unterstützte nur Rudimentär das Beitreten von weiteren Observern zu einer laufenden Sitzung. So war sich zum Beispiel der Inviter der Anwesenheit eines zweiten Observers bewusst, jedoch nicht die anderen Teilnehmer.

Saros war nun so zu erweitern, dass neue Teilnehmer vollständig in eine Sitzung integriert werden.

Planung:
Zuerst war festzustellen, welche Funktionalitäten fehlen, um Multi-Observer vollständig zu unterstützen. Folgende Hauptschwerpunkte traten hierbei auf:

1. Neue Observer-Sessionmitglieder waren nur dem Einladenden bekannt.
   Hier muss ein Mechanismus implementiert werden, um einen neuen Teilnehmer allen Sitzungsteilnehmern mitzuteilen.
2. Markierungen der Textauswahl von Observern werden in der gleichen Farbe dargestellt. Jedem Teilnehmer soll nun eine eigene Farbe zugeordnet werden.
3. Übertragene Markierungen bleiben nach dem Verlassen des dazugehörigen Benutzers aus einer Session bestehen. Markierungen müssen beim Verlassen eines Teilnehmers entfernt werden.

Umsetzung:

1. Nach erfolgreicher Einladung eines neuen Teilnehmers, sendet der Einladende eine Liste aller Sitzungsteilnehmer an den neuen Teilnehmer. Dieser nimmt die weiteren Sessionmitglieder in seine Liste auf und sendet ein Paket mit Join-Extension an diese, um sich bei jedem bekannt zu machen.

2. Die feste Farbzuweisung für einen Driver und einen Observer wurde für mehrere Observer erweitert. Den Teilnehmern einer Sitzung werden lokal Farben zugewiesen.
   Siehe folgender Abschnitt (3.5).

3. Annotations werden mit einem neuen Attribut versehen um die Zuordnung zu einem Mitglied (die Quelle) zu ermöglichen.
   Siehe folgender Abschnitt (3.5).

3.5 Annotations, Farben

Erklärung:

Mit der Unterstützung von mehr Sitzungsteilnehmern trat die bisherige Farb- und Annotation-Funktionalität an seine Grenzen:

1. Die farbliche Markierung für Teilnehmer einer Sitzung benötigte eine Erweiterung, um mehr Teilnehmer zu verwalten.
2. Farben sollen konfigurierbar sein
3. Verlässt ein Teilnehmer die Sitzung sollen seine Markierungen (beim Driver auch Contributions) entfernt werden.
4. Zugewiesene Farben sollen dem jeweiligen Nutzer visuell zugeordnet werden können.

Planung:

1.+2. Um die Konfigurierbarkeit der Farben von Eclipse aus zu erhalten, werden weitere Annotation-Spezifikationen hinzugefügt. Saros soll diese für die Sitzungsmitglieder verwenden.

3. Die Annotations müssen erweitert werden um die Quelle zu speichern.
4. Um die Zuordnung ersichtlich zu machen, soll im SessionView den Personen die entsprechende Farbe zugewiesen werden.

Umsetzung:

1.+2. Um die Markierungsfarben anderer Teilnehmer zu erweitern und konfigurierbar zu halten, wurde die Plugin-KonfigurationsDatei (plugin.xml) um weitere Specifications der Extension „`org.eclipse.ui.editors.markerAnnotationSpecification`“ erweitert. Diesen wurde jeweils eine eigene Standardfarbe zugewiesen.

Von Vorteil ist, dass einzelne Konfigurationen vom Optionsdialog aus Eclipse angepasst werden können und bei späteren Sitzungen wieder zur Verfügung stehen. Nachteilig ist, dass die „plugin.xml“ für hohe Teilnehmerzahlen einmalig manuell erweitert werden muss. Ansonsten wird auf eine einheitliche Markierungsfarbe zurückgegriffen. In einer Sitzung wird jedem Teilnehmer eine Farb-ID zugewiesen. Bei der Erstellung einer Annotation als Wiedergabe einer Auswahl dieses Teilnehmers, wird diese Annotation mit dem

Typ seiner Farb-ID erzeugt und die Annotation der UserID (seiner Jabber ID) zugewiesen. Somit stellt Eclipse diese Annotation in der für diese zugewiesene Farbe da.

3. Um Annotations mit dem Merkmal ihrer Quelle (zu wem diese gehört) auszustatten wurde die Klasse Annotation erweitert (zu AnnotationSaros) um das Attribut Quelle (String) zu verwalten. Auf diese Weise kann jede Annotation später dem User zugeordnet werden.
   Da alle erzeugten Markierungen (genauer Annotations) in einer Liste gespeichert werden, und diese ihre Quelle kennen, kann die Markierung für einen bestimmten Teilnehmer problemlos ermittelt und entfernt werden.
   Dies wird beim Erhalt einer Leave-Nachricht für den entsprechenden Teilnehmer durchgeführt. Hierfür wurde die Funktion `EditorManager::removeAllAnnotations` um einen Parameter erweitert, über welchen eine User-ID übergeben werden kann, um nur die Annotations dieses Teilnehmers zu entfernen.

3.6 GUI für das Einladen von (mehreren) Observern

Erklärung:
Das Einladen von Nutzern war bisher nur einzeln möglich. Mit der Implementierung von Multi-Observern wurde ein benutzerfreundlicher Weg notwendig um mehrere Nutzer in einem Schritt einzuladen.

Plan:
Ein Dialog soll das Auswählen von Kontakten aus dem Roster ermöglichen, welche selektiert und über die Betätigung einer Schaltfläche eingeladen werden sollen. Anschließend soll der Verlauf der Invite-Prozesse dargestellt und die Möglichkeit des selektiven Abbruches einzelner Prozesse geschaffen werden.

So wurde der Nachteil aufgehoben, dass beim Einladen weder Feedback noch eine Abbruchmöglichkeit vorhanden war, solange der Eingeladene nicht auf die Einladung reagierte.

Durchführung:
Zu Beginn stand die Überlegung, ob ein Dialog oder Wizard als Benutzeroberfläche für das Einladen zu nutzen sei. In Hinblick auf leichte Erweiterbarkeit für neue Funktionen und Abfragen entschied ich mich für einen Wizard.

Die zur Einladung möglichen Kontakte (Buddys die gerade online sind und nicht Teilnehmer der aktuellen Sitzung) wurden in einer ersten Implementierung in einer `List (SWT Widget)` dargestellt. Da hier Statusinformationen des Einladeprozesses nicht vernünftig angezeigt werden konnten, wurde stattdessen ein Table genutzt.

Weiterhin wurden Änderungen in der Klasse `Outgoinginvite` notwendig, da die bisherige Fortschrittsdarstellung ersetzt werden sollte. Die Nutzung des `ProgressMonitors` von Eclipse für das Senden der Dateien wurde entfernt, da diese Information durch die neue Oberfläche für alle Einladungen dargestellt wird.

Beim Testen von abgebrochenen Downloads trat ein alter Fehler zutage, welcher dazu führte, dass trotz Abbruches einer Einladung die Projektdateien weiter bis zum Ende übertragen wurden. Dies wurde korrigiert, so dass die Sende-Queue beim Abbruch geleert wird.

Nach Beendigung der Implementierung zeigte sich, dass die Funktionalitäten eines Wizards nicht genutzt werden und die Oberfläche des Wizards wie ein Dialog benutzt wird. Daher entschied ich mich zwecks sauberer Klassennutzung, keinen Wizard sondern lediglich einen Dialog als Grundlage der Oberfläche zu nutzen.

3.7 Robustheit, Netzwerk

Erklärung:
Saros konnte nicht kontrolliert mit Netzwerkfehlern oder Verbindungsverlust umgehen.

Diese Funktionalität soll nachgereicht werden. Dazu gehört das Feststellen von Netzwerkfehlern, als auch der kontrollierte Umgang von Eingaben in dieser Situation.

Planung:
Smack bietet erst mit Version 3.x die Möglichkeit der automatischen Wiederverbindung bei Verbindungsverlust an. Da Saros jedoch im Fehlerfall selbstständig weitere Aktionen durchführen muss um den synchronen Zustand der Projekte sicherzustellen, soll Saros die Erkennung von Verbindungsabbrüchen und deren Behandlung selbst durchführen.

Nach Verbindungsabbruch wird nebenläufig, periodisch versucht, die Verbindung wiederherzustellen. Bis dies erfolgreich geschieht werden Nachrichten und zu sendende Dateien in eine Warteschlange gestellt. Sobald die Jabber-Verbindung wieder aufgebaut werden konnte, werden aufgestaute Nachrichten und Dateien gesendet.

Bei Dateien musste darauf geachtet werden, dass nicht Dateien im geänderten Zustand gesendet werden, da die übertragenen Änderungsaktivitäten doppelt erscheinen würden.

Umsetzung:
Das Bemerken von Verbindungsabbrüchen wird über den `ConnectionListener` von Smack (XMPP) realisiert. Tritt ein Verbindungsfehler auf, wird ein Reconnect versucht. Schlägt dieser fehl, wird Saros in einen Fehlermodus versetzt, welcher weitere Aktionen in Queues speichert und nach Wiederverbindung sendet. Es wird alle 5 Sekunden versucht, die Verbindung wiederherzustellen.

Vermisst ein Teilnehmer Ereignisse anhand des Timestamps, fragt er fehlende Timestamps von den anderen Teilnehmern ab. Hierzu wurde eine neue Nachrichten-Extension (`RequestForActivityExtension`) eingeführt, welche nach einem Timestamp und optional allen folgenden fragt.

3.8 Eclipse 3.3 Inkompatibilität

Erklärung:

In den Vorbereitungen zur ersten Demovorführung am 12.7.2007 zeigte sich, dass mit der gegenwärtig aktuellen Version von Eclipse (v3.3 Europe) das Saros Plugin von Eclipse nicht eingebunden wurde. Dieses Problem zu untersuchen und beseitigen wurde zum nächsten Treffen als Aufgabe formuliert.

Lösung:

Es zeigte sich, das Eclipse das Plugin zwar erkennt und lädt aber nicht startet. Interessanterweise konnte Saros problemlos mit der SDK Version von Eclipse 3.3 starten.

Nach Untersuchungen der Logfiles und Installationsversuchen von der Updateseite zeigte sich die Vermutung bestärkt, dass Abhängigkeiten von Plugins oder Eclipse-Komponenten existieren, die im Eclipse-Release nicht vorhanden waren. Das „Eclipse IDE for Java Developers“ enthält nicht alle Plugins, die in der SDK-Fassung enthalten sind. Da auch verschiedene „Plugin Development Features“ fehlen, kann Saros nicht mit der einfachen Eclipse-Version entwickeln werden.

Zur Ausführung von Saros sollte jedoch nicht die SDK Version notwendig sein. Weitere Untersuchungen zeigten, dass zwei Abhängigkeiten in Saros festgelegt wurden, welche auf SDK-Features verwiesen, aber zur Ausführung von Saros nicht notwendig waren:

- org.eclipse.jdt.source
- org.eclipse.jdt.doc.isv

Umsetzung:

Nach Anpassung der Build-Konfiguration – dem Entfernen der unnötigen Abhängigkeiten - war es nun möglich, ein neu erzeugtes Saros-Plugin in der Java-Developer-Version von Eclipse zu betreiben.

3.9 Fallback für Dateiübertragung

Erklärung:

Tests zeigten, dass es gegenwärtig im FU-internen Netz zu Problemen beim Aufbau von Datentransfers mittels Smack kommt. Chatnachrichten konnten zwischen zwei Clients im Netz ausgetauscht werden, aber der Aufbau einer Dateiübertragung schlägt fehlt, wenn der Verbindungsaufbau zwischen dem WLAN-Netz und dem PCPOOL-Netz aufgebaut werden soll.

Die aktuelle Smack-Implementierung schafft es weder die direkte Transferverbindung herzustellen, noch in den Fallbackmodus zu fallen, um die Datei über den Server zu versenden.

Planung:

Ein manueller Fallbackmechanismus soll auch in dieser Netzwerksituation den Betrieb von Saros ermöglichen. Daher soll eine Option angeboten werden, Saros von Direktverbindungen abzuhalten. In diesem Fall sollen Dateiübertragungen in Nachrichten über den Server übertragen werden.

In Vorbereitungen testete ich, ob Server große Nachrichten akzeptieren. Hier zeigte sich, dass jabber.org bei Nachrichten größer ~50kB die Verbindung trennte. Andere Server wie jabber.cc akzeptieren auch Nachrichten größer 1MB.

Durchführung:

Soll statt eines regulären (Smack) Filetransfers eine Ressource über Nachrichten gesendet werden, übernimmt die Funktion `sendChatTransfer` den Versand. Hierzu wird die Datei eingelesen, BASE64 kodiert und in Chunks einer konfigurierbaren Größe unterteilt. Diese werden als Extension an Chatnachrichten gehangen und an den Empfänger gesendet. Auf der Empfängerseite werden diese Chunks wieder zusammengesetzt und dekodiert.

Nun war interessant, ob es mit dem neuen Mechanismus zu Zeiteinbußen bei der Übertragung von Projekten kommt. Hierfür testete ich die Übertragung des Saros-Projektes (138 Dateien, 24 Verzeichnisse, 1779.3kB) im LAN zwischen einem Desktop-Rechner (LAN) und Notebook (WLAN) an einem Router.

Server	Dauer Dateitransfer	Dauer Chattransfer
Jabber.org	111s	55s
Jabber.cc	59s	45s

Erstaunlicherweise war dieser Versand schneller als der Transfer über Smack. Dies wird vor allem daran liegen, dass der Smack-Transfer für jede Ressource eine Verbindung aushandeln muss, was für jede Datei zu Verzögerungen führt. Dies zu Verbessern ist ein unabhängiger Punkt, welcher zu einem deutlichen Tempogewinn führt.

4. Abschlussbemerkungen

Die Einarbeitung in die Arbeitsweisen von Saros war Anfangs aufwändiger als vermutet. Trotz der Dokumentation war der modulare Aufbau und die vernetzten Klassen und Interfaces anfangs schwer zu durchblicken.

Es wurden viele Fehler korrigiert und Erweiterungen implementiert. Dennoch bleibt noch viel Raum für Verbesserungen und Funktionalitäten. Zum Teil habe ich mit rudimentären Problemen zu kämpfen gehabt, Fehler und Schwächen im alltäglichen Einsatz zu begegnen und auszumerzen.

4.1 offene Punkte

Auch nach meiner Arbeit an Saros gibt es noch immer Potential für Verbesserungen. Die folgende Liste wurde bei den letzten beiden Sitzungen erarbeitet und geben Betätigungsfelder für weitere Studien-/Diplomarbeiten.

- Bei einer bestehenden Jabber-Verbindung und anschließender Konfigurationsänderung sollte die Verbindung abgebrochen und vom Saros neu aufgebaut werden
- Im Kontext-Menü von Buddies sollte der Eintrag "Add new Buddy" eingefügt werden
- Wird eine Person im Session Tab markiert, ist seine Farb-Markierung nicht mehr sichtbar
- Wenn die Internet-Verbindung abbricht, wechselt die Driver-Rolle nicht, da sich der Driver nicht explizit abgemeldet hat (Demo: 3.7.1 Abs. 5)
- Verbindungsüberprüfung nach Rechner-Standby ohne manuelles reconnect
  (zum Beispiel wenn dem Rechner der Strom ausgeht)
- Point of View Farbe des Drivers sollte dieselbe Farbe wie der Driver haben
- „Take-Driver-Role“ funktionierte nicht in der Demo

Weiterentwicklung:

- Möglichkeit für Multi-Chat
- Multi-Driver Unterstützung
- Project-Owner sollte einen Observer entfernen können
- Abbruch der Synchronisation durch Project-Owner sollte möglich sein
  (bei Filetransfer via Chat, beim normalen Filetransfer bereits existent)
- Von Smack 2.x auf 3.x wechseln (?) inklusive Saros-Anpassungen
  - ➔ Vorteil: (neue Features), Timeout bei Filetransferaufbau und so automatischer Wechsel zum Fallback-Filetransfer-By-Chat
- Eventuell den Transport-Manager ersetzen
  - ➔ neue Entwicklungen im Umfeld von ECF analysieren
- auffällige Meldung über Driver-Wechsel (zum Beispiel nicht unterbrechendes Popup),
  siehe Bugtracker # 1619592
- Wenn der Observer den Schreibschutz der Datei aufhebt und etwas ausschneidet
  - ➔ Inkonsistenz im System
  - ➔ Möglichkeit: Datei neu synchronisieren
- Ein- / Ausblenden der Contribution-Farbe nach bestimmten Zeiträumen oder
  Anwendungsfällen oder auf Anwenderwunsch
- Farbeindeutigkeit für Teilnehmer unter allen Beteiligten
- Farbdefinitionen sind standardmäßig für 5 weitere Nutzer definiert. Weitere Nutzerfarben sind manuell vom Anwender in der Plugin.xml zu definieren. Geht das auch programmgesteuert?

5. Verlauf

November 2006 _____

- Einarbeitung in Projekt, Funktionalität studiert
- Dokumentationsstudium (Riads Diplomarbeit)
- Nutzen und Testen von Saros im gegebenen Zustand

Dezember 2006 _____

- Eclipse-Pluginprogrammierung eingearbeitet
- Treffen mit Riad, Saros benutzt, Fragen und Antworten
- Bugfix: Warnung für Observer bei Änderungsversuch als Statusmeldung, statt Messagebox
- Bugfix: Neue Chatnachrichten stehlen nicht mehr den Fokus
- Verbindungsabbrüche prüfen, Konzept zur Behandlung überlegen, Code lesen

Januar 2007 _____

- Homepagegerüst erstellt unter http://dpp.sourceforge.net/
- Inhalte eingefügt (englisch)
    - Overview (teils aus Thesis)
    - Screenshot
    - Anleitungen
    - Technik (todo)
    - Update-Site (todo, mit geeigneter Betaversion einführen)

Februar 2007 _____

- Feature: Verbindungsabbruch erkennen, XMMP-Connection Listener implementiert, ConnectionState.ERROR Status eingeführt und für GUI beachten
- Bugfix: automatisches Reconnecting implementiert
- Feature: Aktionen während Verbindungsfehler aufzeichnen und bei Reconnect senden
- Problematisch: dezentrale timestamp-Generierung

März 2007 _____

- Verbindungsfehler+Aktionsaufzeichnung in verschiedenen Szenarien testen
- Bugfixing, Synchronisierung nach Reconnect verbessert
- Implementiert: Anfordern von fehlenden Aktionen (für fehlende Timestamps) von Peers

April 2007 _____

- Testen des Einladens mehrerer Observer, Suchen von Fehlverhalten und notwendigen Erweiterungen
    - Kein Join-Abgleich, nur der Driver kennt neue Mitglieder, nicht andere Peers
    - Markierungen identisch für verschiedene Mitglieder
    - Asynchrone Reihenfolge der Teilnehmer. Ausschied des Drivers kann somit zu mehreren Driver-Beförderungen führen
    - …
- Bugfix: Bekanntmachung eines neuen Teilnehmers an alle; durch Übertragung der Teilnehmerliste. Observer nehmen neues Mitglied in Reihenfolge auf.

Mai 2007 _____

- Annotations eingelesen, Farbdeklaration in plugin.xml für jeden Annotationtyp
- Untertypen für Selection-Annotations eingefügt.
    - Nachteil: für mehr als X(vordefinierte) Sitzungsteilnehmer, muss (einmalig) die plugin.xml manuell erweitert werden
    - Vorteil: Farben können über Eclipse-Preferences angepasst werden
- Feature:
    - verschiedene Nutzer erhalten zuweisbare, unterscheidbare Farben für Markierungen/Annotations
    - Selection-Annotation-Typ je nach Benutzer setzen
    - Invitation-Wizard implementiert
    - Unschlüssig, ob als Dialog oder Wizard zu implementieren, Wunsch auf Erweiterbarkeit, …? Am Ende erneut evaluieren und eventuell zum Dialog übergehen
    - Auswahl-Kontaktliste als Liste implementiert
    - Progress/Status der Einladung im OutgoingInvitationProcess erweitert, dafür IProgressMonitor entfernt, Progress wird im Wizard dargestellt

Juni 2007 _____

- Invitation Dialog verbessert:
  - Kontaktliste durch Tabelle dargestellt, um weitere Informationen der Einladungen geordnet in Spalten darzustellen
  - Usability verbessert, Dialogelemente ausgrauen welche in aktueller Situation nicht möglich sind
  - Weiter-Button vom Wizard entfernt, Dialog-Implementierung wird sinnvoller
- Bugfix: Nicht gespeicherte Änderungen in Dateien werden beim Einladen nicht übertragen
  - Meldung vor Einladung auf geänderte Dateien und Speichern anbieten
  - Tests zeigten, dass dies mit Eclipse 3.3 nicht mehr funktionierte, da IProject.members() nicht mehr alle Projektdateien zurück liefert. Eclipse Version 3.2 funktionierte. (todo: untersuchen)
- Feature:
  - Quellenmanagement von angewendeten Annotations
    - Anwendung von Annotations mehrerer Anwender möglich; inklusive Löschen aller Annotations eines Nutzers
  - Bugfix: Ausscheidender Observer, Driver/Driverwechsel hinterlässt nun keine fehlerhaften Annotations, sie werden quellenabhängig gelöscht

Juli 2007 _____

- Saros Updatesite Mechanismus angeschaut, es fehlen mir noch Details zum Aufsetzen auf Projekthomepage, notfalls Riad fragen
- Bugfix: Nach Abbruch einer Einladung während der Synchronisierung wurden weiterhin Dateien übertragen und es kam zu einer Exception. Übertragung wird nun beendet und Transferqueue geleert
- ECG
  - Alte, inzwischen ungültige, URL aus Doku (Thesis) http://www.electrocodeogram.org verwirrte, bis ich Projektseite auf berliOS (http://developer.berlios.de/projects/ecg/) fand
  - Heruntergeladen, einlesen…
  - Idee:
    - ECG Konfigurationsdatei von Saros aus anpassen, welche Übertragungsdetails regelt
    - ECG Projekt einarbeiten, erweitern
- Demovorführung, Treffen mit den Betreuern am 12.7.2007
  - Dateitransfer scheiterte im Institutsnetz an Sicherheitsvorkehrungen im WLAN Netz

August 2007_____

- Dateitransfer im Institutsnetz getestet
    - Smack kann keine Verbindung aufbauen, sobald das WLAN-Netz beteiligt ist
    - Anfrage an die Technik bestätigte Restriktionen beim Übergang zum WLAN-Netz
- Fallbackmechanismus für Dateiübertragung implementiert
    - Übertragung optional (konfigurierbar in erweiterten Saros-Einstellungen) via (Chat-)Nachrichten
    - Aufteilung bei großen Dateien (ebenfalls konfigurierbares Limit)
- Demovorführung, Treffen mit den Betreuern am 3.8.2007
    - Testliste erfolgreich durchgeführt, eine Reihe von Feature-Requests und kleineren Bugs siehe Abschnitt 4.1
- Homepage vervollständigen
- Updateseite einrichten
- Studienarbeit-Dokumentation fortführen