Freie Universität Berlin

Fachbereich Mathematik und Informatik

Institut für Informatik

Diplomarbeit

Evaluation und Überarbeitung der Usability von Saros

Alexander Waldmann

Berlin, 11.04.2012

Gutachter

Prof. Dr. Lutz Prechelt

Institut für Informatik

Freie Universität Berlin

Betreuerin

Julia Schenk

Institut für Informatik

Freie Universität Berlin

Eigenständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Diplomarbeit selbstständig und ohne unerlaubte Hilfe angefertigt habe. Alle verwendeten Hilfsmittel und Quellen sind im Literaturverzeichnis vollständig aufgeführt und die aus den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht.

Berlin, 11.04.2012 Alexander Waldmann

Zusammenfassung

In dieser Diplomarbeit wird die Usability (Gebrauchstauglichkeit) von Saros, einem Werkzeug zur räumlich verteilten Programmierung, evaluiert und überarbeitet. Der Fokus der Analyse lag hierbei auf der Effektivität und Effizienz von Saros zur Bearbeitung von Programmieraufgaben.

Hierzu werden zunächst analytische und empirische Evaluations-Verfahren hinsichtlich ihrer Eignung zur Analyse von Saros bewertet. Bei der anschließenden Durchführung einer heuristischen Evaluation und mehreren Usability-Tests unter Anwendung eines Codiscovery-Learning Ansatzes[1], werden 128 Usability-Probleme identifiziert und katalogisiert.

Davon werden 54 Probleme in einem iterativen Prozess bearbeitet, bestehend aus Entwicklungsarbeiten und darauffolgenden Nutzertests. Für die verbleibenden Probleme werden erste Lösungskonzepte und Prototypen erstellt.

Neben einer Steigerung der Effizienz durch Geschwindigkeitsoptimierungen in der Programmlogik, zeigen die Beobachtungen in den Evaluationen der Prototypen eine Verringerung der verbalen Koordination zwischen den Probanden und eine daraus folgende Verbesserung in der Effizienz bei der gemeinschaftlichen Programmierung mit Saros.

Durch quantitative (standardisierter Fragebogen) und qualitative Befragungen (Interviews) wird abschließend ein Einblick in die Veränderung der subjektiven Zufriedenheit der Probanden gewonnen.

[1] Beim Codiscovery-Learning erkunden jeweils zwei Probanden gemeinsam die Systemfunktionalität und kommentieren dabei laut. Die Beschreibung des Verfahrens findet sich in Abschnitt 3.6.4

Konventionen

- [AutorJahr] kennzeichnet den Verweis auf eine Quelle im Literaturverzeichnis, die anhand der Kurzbezeichnung identifiziert wird.

- [1] verweist auf eine Fußnote am Ende der Seite. Dies wird für Begriffserklärungen verwendet, damit eine aufwändige Suche im Literaturverzeichnis entfällt, wenn dies den Lesefluss stören würde.

- *Kursive Schrift* wird im Fließtext für Java-Klassennamen verwendet - unter Abbildungen für die Bildunterschrift.

- Code-Beispiele sind eingerahmt

- Sofern bei einer Abbildung keine explizite Quellenangabe vorhanden ist, handelt es sich um einen selbstständig angefertigten Screenshot (Bildschirmabbild) oder um eine Grafik, die eigenständig auf Basis der erhobenen Daten erstellt wurde.

Inhaltsverzeichnis

1 Einleitung ... 6
1.1 Begriffsdefinitionen ... 8
2 Vorstellung von Saros ... 12
2.1 Entwicklungsgeschichte und vorangehende Arbeiten ... 15
2.2 Einblicke in die tatsächlichen Nutzung ... 16
3 Evaluation der Usability ... 20
3.1 Verfügbare Ressourcen ... 20
3.2 Geeignete Analyseverfahren ... 21
3.3 Heuristische Evaluation ... 23
3.4 Cognitive Walkthrough ... 26
3.5 Pluralistic Walkthrough ... 27
3.6 Usability Tests ... 27
3.7 Interviews & Fragebögen ... 34
3.8 Systemprotokolle, Nutzer-Feedback ... 35
3.9 Verfahren zur Katalogisierung der Probleme ... 37
4 Durchführung der Analyse ... 42
4.2 Zusammenfassung der Analyse-Ergebnisse ... 48
4.3 Reliabilität der Ergebnisse ... 66
4.4 Validität der Ergebnisse ... 67
5 Verbesserung der Usability ... 72
5.1 Arbeitspakete ... 73
5.2 Prototyp: Changelog für Saros-Sitzungen ... 104
5.3 Ergebnisse der Evaluation der Überarbeitung ... 108
5.4 Auswertung der Fragebögen zu den Prototypen ... 109
6 Fazit ... 111
6.1 Rückblick auf die Nutzungsstatistik ... 114
6.2 Vorschläge zur Verbesserung der Prozesse ... 114
7 Anhang ... 117
7.1 Übersicht der Usability Methoden von Nielsen ... 117
7.2 Die12 Heuristiken von Sarodnick und Brau ... 117
7.3 Fortschrittsanzeigen in Eclipse ... 119
7.4 Auswertung der Fragebögen ... 124
7.5 Kontextmenü-Strukturen nach Microsoft Guideline ... 124
7.6 Die Arbeit im Saros-Team ... 125
7.7 Der Usability-Problemkatalog ... 129
8 Quellenverzeichnis ... 173

1 Einleitung

Gegenstand dieser Diplomarbeit ist die Analyse und Überarbeitung der Usability[2] von Saros. Saros ist ein Werkzeug zur räumlich verteilten Software Entwicklung, das als ein Plugin[3] für die Entwicklungsumgebung Eclipse[4] entwickelt wurde. Es wird durch die Arbeitsgruppe Software Engineering an der Freien Universität Berlin, in der Programmiersprache Java, kontinuierlich weiterentwickelt.

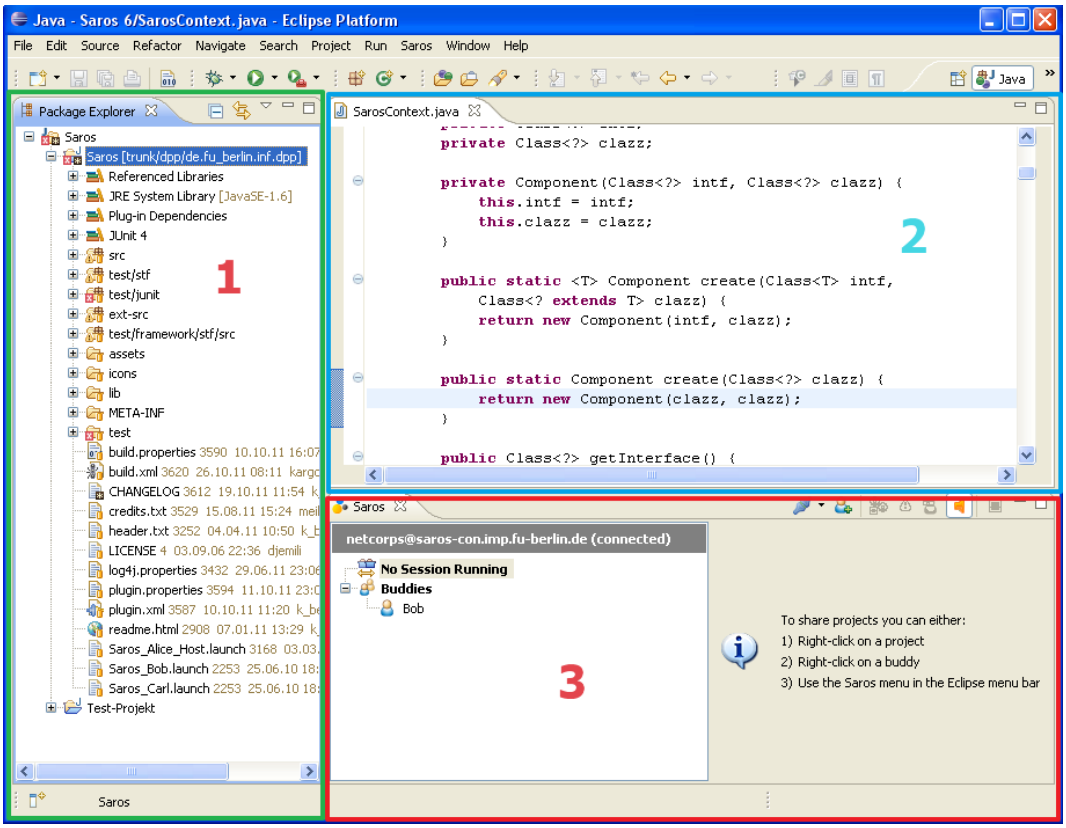

Abbildung 1: Eclipse Entwicklungsumgebung mit Package Explorer (1), dem Editor-Bereich (2) sowie dem Saros-View (3) mit der Buddyliste (links).

Saros wurde entwickelt, um verteilte Programmierung zu erleichtern[5]. Daraus ergeben sich besondere Anforderungen an die Struktur des Programms und die Informationen, die bereitgestellt werden. So nehmen Nutzer bei der normalen Paarprogrammierung (in räumlicher Nähe, vor einem einzelnen Computer) automatisch und oft unbewusst wahr, welche Dateien der Andere gerade geöffnet hat, welcher Teil der Datei einsehbar ist und wie konzentriert der Andere ist. Diese Informationen fehlen bei räumlich verteilter Zusammenarbeit und sollten daher von Saros übertragen werden, da sie zur Reduktion von Unsicherheit und

---

[2] Usability kann mit „Gebrauchstauglichkeit" übersetzt werden. Genauere Definition siehe Abschnitt 2.1 Begriffsdefinitionen > Usability

[3] Softwaremodul, das in einer Applikation eingebunden läuft und über definierte Schnittstellen mit dieser kommuniziert

[4] www.eclipse.org

[5] www.saros-project.org

zur spontanen Koordination für eine effektive und effiziente Zusammenarbeit wichtig sind (Vgl. [Gutwin1996], [Gutwin2004]).

Das Plugin wurde kontinuierlich im Funktionsumfang erweitert, der Webseite nach allerdings bei gleichzeitiger Vernachlässigung von Benutzbarkeit und emotional-ästhetischen Aspekten. So sind die Rückmeldungen der Anwender über die Saros-Nutzerumfrage vor allem von Frustration über Probleme bei der Nutzung geprägt, darunter wiederholte Kritik über fehlende Rückmeldungen über die Systemaktivitäten. Vergangene Arbeiten und Tests haben zudem ergeben, dass zu wenig Awarenessinformationen[6] übermittelt werden, und die Präsenz deutlich hinter der klassischen, nicht verteilten, Paarprogrammierung liegt (Vgl. [Solovjev2011]). Auch wurde in diesem Zusammenhang der Einladungsprozess zum Aufbau einer Saros-Sitzung als kompliziert und wenig zusammenhängend empfunden.

Um das Plugin verbessern zu können, werden zunächst die Usability-Probleme in einer Analyse identifiziert, wobei ein Katalog von Usability-Problemen aufgebaut wird, der dann als Grundlage für die Überarbeitung dient.

Im Gegensatz zu vorherigen Arbeiten, in denen unter anderem die Out-of-Box-Experience von Saros mit Nutzern, die das System noch nicht kennen geprüft und verbessert wurde, oder Ansätze ausgearbeitet wurden, wie die User Experience in Saros verbessert werden kann, wird in dieser Arbeit die Usability bei wiederholtem Einsatz von Saros zur verteilten Programmierung, geprüft und verbessert.

Bei der Analyse werden hierbei Verfahren zur analytischen Evaluation und zur empirischen Evaluation kombiniert, um eine Entscheidungsgrundlage für die Priorisierung der Usability-Probleme zu haben.

Nach einer heuristischen Evaluation werden unter realen (nicht Labor-) Bedingungen Tests mit Nutzern durchgeführt. Die Nutzer lernen Saros in Pilot-Tests kenne, und erledigen damit zum Teil Arbeitsaufgaben (im Rahmen ihrer Tätigkeit als Angestellte in einem Software-Unternehmen), und zum Teil vorgegebene Programmieraufgaben, um den Verlauf der Tests besser kontrollieren zu können.

Bei der folgenden Verbesserung der Usability, auf Basis der Analyseergebnisse, ist das Ziel, eine möglichst deutlich wahrzunehmende Verbesserung herbeizuführen. Hierbei werden die Probleme nach ihrer Problemschwere priorisiert, und Lösungsansätze für die zu bearbeitenden Probleme ausgearbeitet und implementiert. Die Wahl der zu behebenden Probleme erfolgt dabei unter Berücksichtigung einer Einschätzung der Behebbarkeit in der vorgegeben Zeit.

Änderungen an der Oberfläche werden iterativ durchgeführt, um zu prüfen, ob durch die Veränderungen neue Usability-Probleme entstanden sind, und ob die

---

[6] „Definition von Awareness nach Gutwin und Greenberg (1995, übersetzt): Es handelt sich um auf die Minute aktuelles Wissen um die Aktivitäten anderer Menschen, das vom einzelnen benötigt wird, um sich mit diesen zu koordinieren und die eigenen Aufgaben innerhalb der Gruppe zu erfüllen.“ [SarosWeb2]

Probleme die behoben werden sollten, nicht mehr auftreten. Phasen aus Implementierung und Evaluation in weiteren Usability-Tests wechseln sich hierbei ab.

Sowohl nach den Tests mit dem Original-Saros als auch nach den Tests mit den entwickelten Prototypen werden die Nutzer über Interviews und Fragebögen nach ihrer subjektiven Einschätzung von Saros gefragt, um Einblicke in die Veränderung der Nutzerzufriedenheit zu erhalten.

Nach dieser Einführung, mit einer Beschreibung der Motivation für diese Diplomarbeit, werden im nächsten Abschnitt die in dieser Arbeit verwendeten Begriffe definiert und die Funktionalität von Saros wird vorgestellt.

1.1 Begriffsdefinitionen

Um zu verstehen, was im Rahmen dieser Arbeit analysiert und verbessert wurde, und inwiefern dies von Bedeutung ist, wird in diesem Abschnitt zunächst der Begriff der Usability vorgestellt.

Über die Begriffsdefinition und geeignete deutsche Übersetzungen gibt es zahlreiche Quellen. Usability als Begriff klar zu definieren, bzw. eine eindeutige, deutsche Übersetzung zu finden, ist aufwändig. Bei der Recherche nach einer leicht verständlichen Definition fällt auf, dass es neben der „Usability“ weitere Begriffe, wie die „Out-of-Box-Experience“, und „User Experience“ gibt, die im Anschluss kurz definiert werden.

Mögliche Übersetzungen des Begriffs „Usability“ ins Deutsche beinhalten
- Benutzerfreundlichkeit
- Bedienbarkeit / Benutzbarkeit
- Gebrauchstauglichkeit = Benutzbarkeit + Zufriedenstellung [WikiUsability]

Der Begriff der Usability ist hierbei nicht auf Softwaresysteme beschränkt. Nach zunächst sehr spezifischen Definitionen der Usability in Bezug auf Software-Systeme, wurden Normen, wie z.B. die ISO 9241, so weit verallgemeinert, dass sie sich auf „Mensch-Computer Interaktion“ allgemein beziehen und damit auch die Hardware (Tastatur/Maus/Bildschirm) mit einbeziehen. Ein Ansatz für eine allgemein gültige Definition findet sich in dieser Norm, die von der Internationalen Organisation für Standardisierung (ISO) entwickelt wurde:

> "Usability bezeichnet das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und mit Zufriedenheit zu erreichen.“ [ISO9241]

Nutzer verwenden ein System zu einem Zweck – bei Saros zum Beispiel, um räumlich verteilt zu programmieren, gemeinsam mit anderen Programmierern. Dabei beschreibt der Begriff Usability, wie effektiv die von Saros bereitgestellten Mechanismen für die Zielerreichung sind und wie effizient die Arbeit mit dem Werkzeug an sich ist (wenn die Übertragung eines Tastendrucks eine Minute dauern würde, wäre dies beispielsweise hochgradig ineffizient). Auch spielt die Zufriedenheit der Nutzer bei der Verwendung des Systems eine Rolle. Um nun genauer zu ermitteln, wie Effizienz, Effektivität und Zufriedenheit betrachtet

werden können, werden die fünf Kernattribute der Usability, die Nielsen in seinem Referenzwerk „Usability Engineering“ beschreibt, vorgestellt.

- Leichte Erlernbarkeit: Das System sollte so konstruiert sein, das Nutzer sich schnell zurecht finden und mit der Erledigung der eigentlichen Aufgabe beginnen können

- Effizienz der Bedienung: Das System sollte effizient genutzt werden können, sodass nach dem Erlernen des Systems ein hoher Produktivitätsgrad erreicht werden kann

- Leicht zu merken: Das System, dessen Interaktionskonzepte und Bedienoberfläche sollte leicht zu merken sein, sodass auch gelegentliche Nutzer das System verwenden können, ohne die Bedienung jedes Mal neu erlernen zu müssen

- Fehlervermeidend & Fehlerarm: Das System sollte eine geringe Fehlerrate haben, sodass Nutzer weniger Fehler bei der Bedienung des Systems machen, und wenn sie Fehler machen, diese leicht bereinigen können. Katastrophale Fehler dürfen nicht auftreten.

- Zufriedenheit: Das System sollte angenehm zu bedienen sein, sodass Nutzer subjektiv zufrieden sind wenn sie das System bedienen. [Nielsen1993, S.26]

Bei der Analyse von Saros geht es darum, zu prüfen, in wie weit Saros diese Grundsätze befolgt, und Mängel in der Usability zu identifizieren, um diese beheben zu können. Dabei soll die Eignung von Saros zur verteilten Programmierung insbesondere aus Sicht der Nutzer betrachtet werden.

Für die „Usability aus Nutzersicht“, findet sich im Paper von McGee et Al. „Understanding the usability construct: User-perceived Usability“ die Formulierung:

> “Usability is your perception of how consistent, efficient, productive, organized, easy to use, intuitive, and straightforward it is to accomplish tasks within a system.”
> [McGeeRichDumas2004, S.909]

Frei übersetzt: Usability ist deine Wahrnehmung, wie konsistent, effizient, produktiv, organisiert, einfach zu benutzen, intuitiv und direkt es möglich ist, eine Aufgabe mit einem System zu lösen. Hier werden weitere Eigenschaften genannt, die für eine gute Usability entscheidend sind:

- „Intuitiv“ meint, dass das System sich erwartungskonform verhält und so strukturiert ist, wie andere Systeme, die die Anwender bereits kennen
- „konsistent“ beschreibt, dass die Oberfläche einheitlich gestaltet sein sollte, Elemente mit gleicher Aktion auch gleich benannt sind, und das insbesondere anwendbaren Richtlinien gefolgt wird (z.B. der Eclipse Guideline, da es sich bei Saros um ein Eclipse Plugin handelt)

Bei der Betrachtung der Erwartungskonformität eines Systems ist es einerseits wichtig, die Zielgruppe genau zu kennen, andererseits sind Einblicke in das mentale Modell der Nutzer notwendig um zu erkennen wann Abweichungen vorliegen (Vgl. [Nielsen1993]).

Das mentale Modell beschreibt, was ein Nutzer über das ihm vorliegende System glaubt. Ein mentales Modell basiert auf Glauben, nicht auf Fakten. Es ist ein Modell dessen was Benutzer über ein System glauben und wissen (oder meinen zu wissen). Jeder Nutzer hat ein mentales Modell, das sich deutlich von dem mentalen Modell anderer Nutzer und insbesondere auch dem Modell der Designer des betrachteten Systems unterscheiden kann. Auf Basis umfangreichen Wissens über die (technischen) Eigenschaften des gestalteten Systems, formen die Designer ein mentales Modell, was dazu führt, dass diese überzeugt sind, das jede Funktion des Systems leicht zu verstehen wäre. Ohne Wissen über die tatsächlichen Eigenschaften des Systems, ist es wahrscheinlich, dass Nutzer Fehler machen und das Design als schwieriger zu bedienen empfinden (Vgl. [Nielsen2010]).

Eines der Ziele bei der Analyse von Saros ist es, herauszufinden, bei welchen Begriffen und Konzepten in Saros die Nutzer Schwierigkeiten mit dem Verständnis haben. Nutzer gleichen z.B. Beschriftung von Oberflächenelementen, Icons und Strukturen mit anderen, ihnen bekannten Systemen ab. Ist eine Bezeichnung in beiden Systemen gleich, erwarten Nutzer auch ein ähnliches Ergebnis bei Interaktion mit dem Element. Verhält sich das System anders, als es die Nutzer erwarten, müssen sie zunächst darüber nachdenken, wie dieses System nun funktioniert. Ist eine deutliche Diskrepanz zwischen der Erwartungshaltung und dem tatsächlichen Ergebnis einer Interaktion mit dem System gegeben, so kommt es häufig dazu, dass der Anwender bei der Bearbeitung einer Aufgabe mehr Zeit benötigt (Vgl. [Moser2000]), was wiederum in einer Nutzer-Beobachtung auffallen kann. Eine Verzögerung bei der Bedienung des Systems durch Verständnisschwierigkeiten verringert die Effizienz bei der Arbeit mit dem System, und stellt damit ein Problem dar.

Bei der Überarbeitung des Systems, nach der Identifikation der Schwachstellen, sollten Grundsätze für die Gestaltung einer Benutzer-System-Schnittstelle befolgt werden, wie sie beispielsweise in der ISO 9241-110 „Grundsätze der Dialoggestaltung“ definiert werden:

- Aufgabenangemessenheit – geeignete Funktionalität, Minimierung unnötiger Interaktionen
- Selbstbeschreibungsfähigkeit – Verständlichkeit durch Hilfen / Rückmeldungen
- Lernförderlichkeit – Anleitung des Benutzers, Verwendung geeigneter Metaphern, Ziel: minimale Erlernzeit (Vgl. auch [Dix2003])
- Steuerbarkeit – Steuerung des Dialogs durch den Benutzer
- Erwartungskonformität – Konsistenz, Anpassung an das Benutzermodell
- Individualisierbarkeit – Anpassbarkeit an Benutzer und an seinen Arbeitskontext
- Fehlertoleranz – Die Funktionsweise des Systems auch bei unvorhergesehenen Fehlern aufrechterhalten
  [WikiIso9241]

Werden diese Grundsätze ignoriert, oder nicht ausreichend verfolgt, entstehen nach Nielsen (1993) Benutzeroberflächen, die vielleicht noch vom Designer oder Entwickler selbst bedienbar sind, aber für die eigentlichen Nutzer ungeeignet sind.

Neben der Usability des Produkts spielt auch die sogenannte „User Experience“ eine Rolle, die „praktische Erfahrungen und Erlebnisse bei der Benutzung eines interaktiven Systems“ meint [Jetter2006]. Der Begriff wurde ursprünglich von Don Norman im Rahmen seiner Tätigkeit bei Apple geprägt als „die Gesamtwirkung aller bei der Benutzung verwendeten Gegenstände auf den Benutzer [...] – angefangen von der Produktverpackung über die Gestaltung der Interaktion bis zum Grafikdesign auf dem Bildschirm“ [Jetter2006].

Jetter beschreibt, dass die User Experience „nicht nur durch die Effektivität, Effizienz und Zufriedenheit bei der Benutzung von Anwendungen geprägt ist[…], sondern ein ganzheitlicher Begriff von der Interaktion zwischen Mensch und Computer herangezogen wird, der auch die Motivation, Identifikation, Ästhetik oder – in der Sprache mancher Käuferschichten – die „Coolness“ oder „Sexiness“ umfasst.“ [Jetter2006]

Faktoren, wie die optische Gestaltung der Oberfläche, Anordnung und Formen von Bedienelementen spielen eine wichtige Rolle beim Nutzererlebnis (Vgl. [Jetter 2006]). Mit der Einführung von Techniken, wie haptischen Displays (z.B. von Senseg [7]), kann zukünftig sogar die Haptik von Bedienelementen ein bedeutsamer Faktor bei der Gestaltung interaktiver System werden.

Die User Experience beinhaltet damit nicht nur die Usability des Systems, sondern fasst weiter, und bezieht auch emotional–ästhetische Aspekte ein, die bei der Analyse und Überarbeitung von Saros berücksichtigt werden müssen. Für Softwareentwickler (zumindest aus eigener Erfahrung) steht zwar die Effizienz der Arbeit mit dem System im Vordergrund, allerdings können sie sich auch an ästhetisch gestalteten Oberflächen erfreuen, weshalb eine Überarbeitung kosmetischer Mängel ebenfalls Bestandteil dieser Arbeit sein wird.

Neben der Usability und der User Experience gibt es noch einen weiteren Begriff, die sogenannte Out-of-Box-Experience (OOBE), die den Erstkontakt eines Nutzers mit einem System beschreibt. Für die Out-of-Box-Experience sind der Installationsprozess, und die Anlernung des Nutzers an die Bedienkonzepte des Systems entscheidend. Diese Aspekte wurden im Zusammenhang mit Saros bereits in der Arbeit von Björn Kahlert [Kahlert2011] untersucht und verbessert, wobei hier der Fokus auf der erstmaligen Konfiguration (Einrichtung eines Benutzerkontos), dem Hinzufügen von Buddies (Kontakte) und dem Sitzungsaufbau und Sitzungsabbau lag.

Der Erstkontakt der Nutzer mit dem System wird zwar für spätere Auswertungen, durch andere Team-Mitglieder, bei der Durchführung von Pilot-Usability-Tests aufgezeichnet, die Out-of-Box-Experience wird in dieser Arbeit aber nicht weiter betrachtet.

---

[7] http://www.senseg.com/

2 Vorstellung von Saros

Da für die Analyse und die Überarbeitung der Usability von Saros, die Kenntnis des Systems, dessen Funktionsweise und die Verwendungszwecke durch die Zielgruppe nötig sind, folgt eine entsprechende Übersicht in diesem Kapitel.

Saros ist ein Tool für räumlich verteilte gemeinschaftliche Software Entwicklung. Eingebettet als Plugin in die Entwicklungsumgebung Eclipse, synchronisiert Saros Projekte (die enthaltenen Dateien und Ordner) zwischen Teilnehmern einer Saros-Sitzung, sodass alle Teilnehmer innerhalb der Sitzung stets den gleichen Arbeitsstand haben. Die Verwendungszwecke bei der Softwareentwicklung beinhalten der Saros-Webseite[8] zufolge unter anderem

- Gemeinsames Ansehen von Code, um jemandem etwas zu erklären, jemandem um Rat zu fragen oder gemeinsam Mängel zu suchen (Reviews, Walkthroughs)
- Verteilte Paarprogrammierung, d.h. die eng gekoppelte gemeinsame Arbeit an einer gemeinsamen Aufgabe.
- Verteilte Side-by-Side-Programmierung, d.h. die lose gekoppelte Zusammenarbeit, bei der man die meiste Zeit an verschiedenen Aspekten und nur gelegentlich direkt zusammen arbeitet, auch mit mehr als zwei Personen

Saros ermöglicht die verteilte Programmierung, bei der die Teilnehmer sich nicht in räumlicher Nähe zueinander befinden müssen, auch mit mehr als 2 Sitzungsteilnehmern. Nutzer können innerhalb einer Saros-Sitzung unabhängig voneinander Änderungen an synchronisierten Projekten vornehmen, bei Bedarf aber in einen sogenannten „Follow Mode“ schalten, bei dem sie direkt den Änderungen eines ausgewählten Nutzers in einem Beobachtermodus folgen.

Saros stellt dazu innerhalb der Oberfläche von Eclipse, in mehreren Views, seine Oberfläche bereit, mit der Benutzer interagieren können, um Sitzungen aufzubauen, Kontakte hinzuzufügen, eine Voice-over-IP Verbindung aufzubauen, Dateien zu senden, zu chatten und um den Bildschirminhalt zu übertragen.

Benutzer müssen nach der Installation von Saros über die von Eclipse bereitgestellten Mechanismen das Plugin zunächst mit Zugangsdaten zu einem XMPP Account konfigurieren. Die Kommunikation zwischen mehreren Saros-Instanzen erfolgt sowohl über XMPP-Nachrichten[9], als auch über direkte Socket Verbindungen[10]. Saros nutzt zur Kommunikation das XMPP-Protokoll, einem XML-Routing Protokoll, das vor allem für Instant Messaging eingesetzt wird. Saros verwendet eine Serverbasierte Architektur, bei der die Klienten XMPP Nachrichten über XMPP Server versenden.

Die Übertragung von Nutzer-Interaktionen (Tastatureingeben in Editoren, Selektionen, Dateiänderungen) zu anderen Sitzungsteilnehmern erfolgt hierbei

---

[8] www.saros-project.org
[9] Zeichenketten mit XML-Inhalt nach XMPP Standard
[10] Bidirektionale Software-Schnittstelle zur Netzwerk-Kommunikation

durch das Kapseln von Aktivitätsbeschreibungen (Activity-Klassen) in XMPP Nachrichten, die dann an andere Klienten (die Saros-Instanzen der anderen Sitzungsteilnehmer) verschickt werden, wobei der Dateitransfer über direkte Socket-Verbindungen erfolgt.

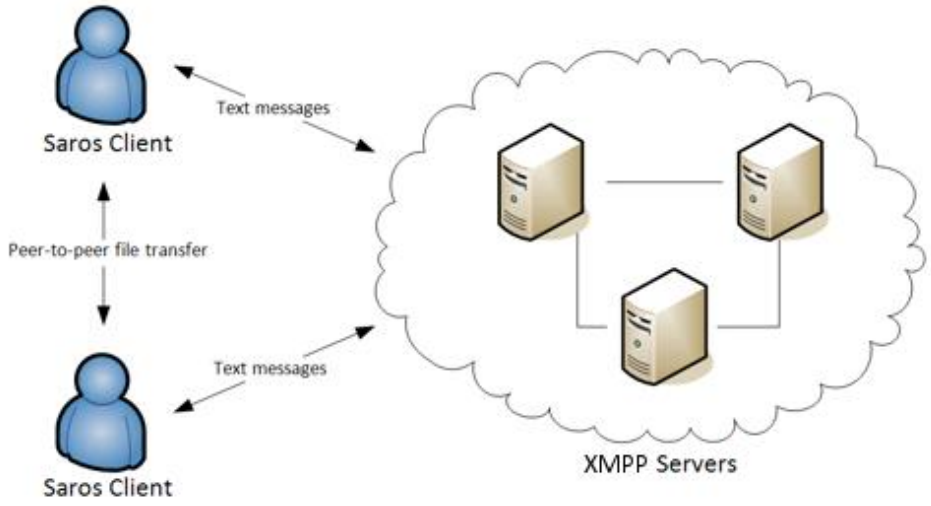

Abbildung 2: Übersicht der Saros-Architektur (Quelle: www.saros-project.org)

Nutzer müssen vor dem Start einer Sitzung zunächst einmalig andere Personen als Kontakte hinzufügen, die wiederum die Kontaktaufnahme bestätigen müssen. Danach sehen beide Nutzer den anderen in der Liste der „Buddies", innerhalb des Saros Views. Nutzer können nun auf unterschiedlichen Wegen eine Saros-Sitzung starten.

Neben dem Menüeintrag „Saros > Share Project(s)" in der Eclipse-Menüleiste, kann der Einladungsprozess über die Kontextmenüs von den Buddies im Saros-View, und über das Kontextmenü von Projekten, Ordnern oder Dateien gestartet werden. Je nach dem welchen Weg der Nutzer geht, liegen Saros zum Aufbau der Sitzung entweder schon alle Informationen zum Starten der Einladung vor, oder aber ein Wizard (ein mehrseitiger Dialog) wird geöffnet der die notwendigen Daten abfragt.

Für den Sitzungsaufbau sind grundsätzlich zwei Informationen notwendig:

- Welche Nutzer sollen in die Sitzung eingeladen werden
- Welche Dateien/Ordner/Projekte sollen Bestandteil der Sitzung sein

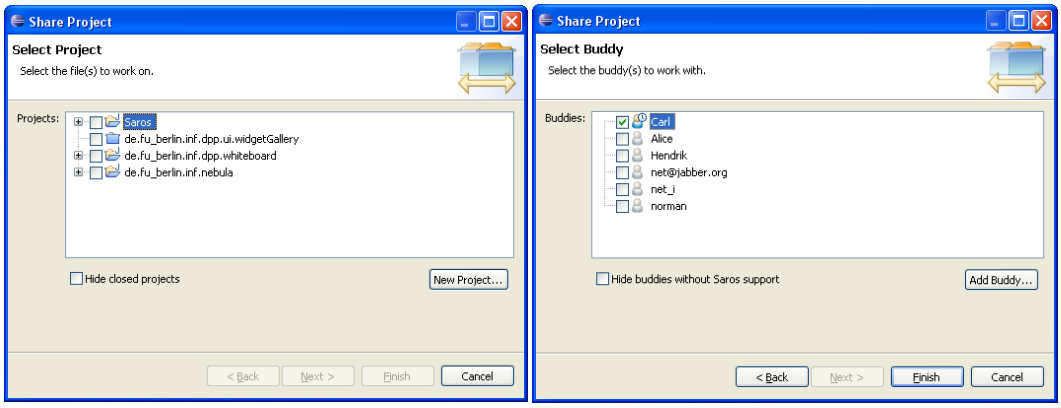

Abbildung 3: Screenshots der beiden Seiten des Einladungs-Wizards (Ressourcen und Teilnehmer)

Nach dem Start einer Sitzung werden die Sitzungsteilnehmer in einer Liste im Saros-View angezeigt. Hierbei sind die Einträge der Nutzer mit den Farben hinterlegt, die Saros zur Verbesserung der Awareness einsetzt um Texteingaben und Markierungen hervorzuheben. Saros erstellt Farbmarkierungen für Texteingaben und Markierungen aller anderen Sitzungsteilnehmer in der gerade offenen Datei, wenn diese Bestandteil der Sitzung ist. Seitliche Farbbalken repräsentieren den für andere Nutzer sichtbaren Teil der Datei.

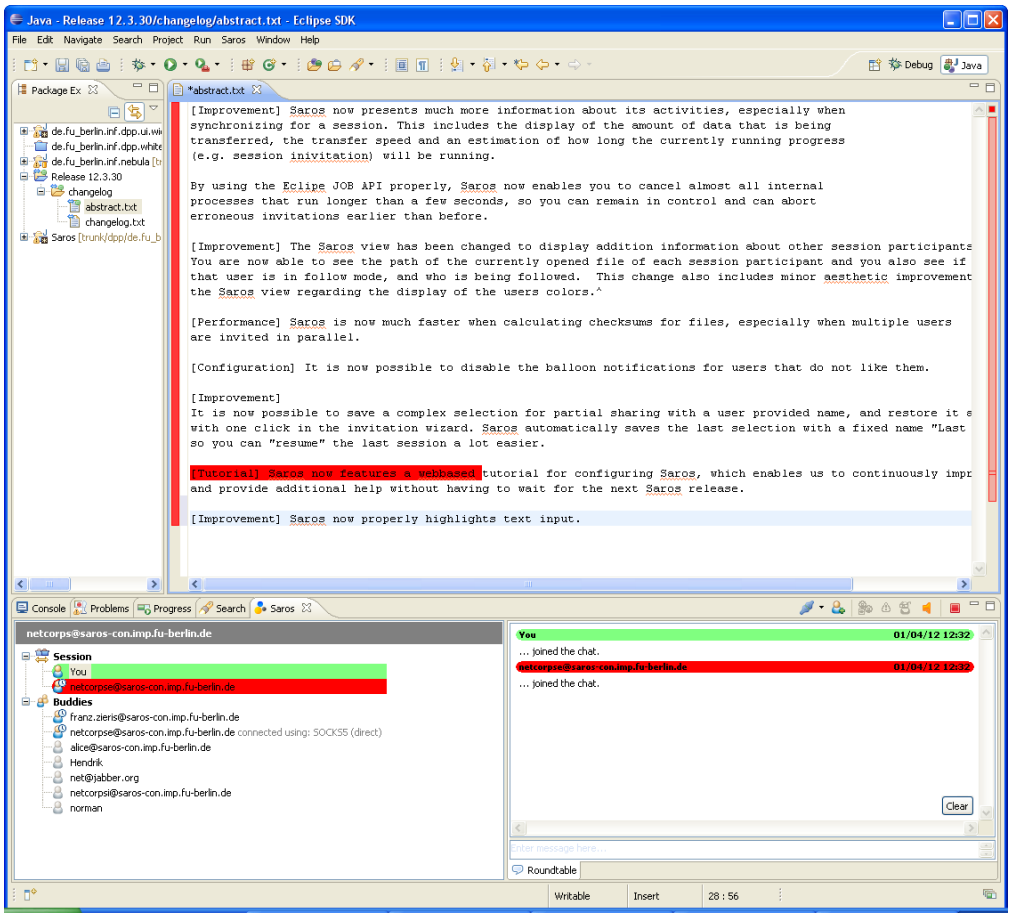

Abbildung 4: Eclipse-Ansicht in einer laufenden Saros-Sitzung mit farbig markierten Änderungen durch einen der Sitzungsteilnehmer

Im sogenannten Follow-Modus kann ein Nutzer wählen, dass er einem anderen Nutzer in der laufenden Sitzung folgen möchte. Saros öffnet dann automatisch Dateien, die dieser Nutzer öffnet, und verschiebt (scrollt) den sichtbaren Bereich der Datei analog zu dem Bereich, den der gefolgte Nutzer einsieht.

Neben der grundlegenden Funktionalität von Saros wurden experimentelle Funktionen hinzugefügt, die instabil implementiert sind und bei denen bereits in der heuristischen Evaluation sehr viele Probleme identifiziert werden. Dies betrifft insbesondere die Funktion für Sprachverbindungen (Voice-over-IP, VoIP), und das Screen-Sharing, bei dem der komplette, oder ein Teil des Bildschirminhalts, an andere Nutzer übertragen wird. Eine grundlegende Analyse der Funktionen in der heuristischen Evaluation ist erfolgt, und einige Notizen aus der Beobachtung von Videoaufzeichnungen aus einer vorangehenden Bachelorarbeit [Solovjev2011] wurden angefertigt, um den Zustand dieser Funktionen zu verdeutlichen. Allerdings wurden diese Funktionen in den Usability-Tests nicht verwendet, da hier zunächst eine grundlegende Überarbeitung notwendig wäre, bevor aufwändige Nutzertests zu rechtfertigen wären.

2.1 Entwicklungsgeschichte und vorangehende Arbeiten

Um zu verstehen, wie es zu der Vielzahl an Usability-Problemen in Saros kommen konnte, und welche Datenbasis dieser Arbeit zur Verfügung stand, erfolgt in diesem Abschnitt eine Betrachtung der vorangehenden Abschlussarbeiten.

Die anfängliche Entwicklung von Saros begann 2006 im Rahmen der Diplomarbeit von Riad Djemili [Djemili2006], der „Eine Eclipse-Erweiterung zur verteilten Paarprogrammierung“ entwickelte.

Das System wurde schrittweise um die Möglichkeit erweitert, mit mehreren Nutzern gleichzeitig Projekte zu verändern, sowie der Möglichkeit, in einer Sitzung an mehreren Projekten arbeiten zu können, erweitert.

Die Usability des Systems rückte spätestens mit der Bachelorarbeit von Hoffen (2010) etwas mehr in den Vordergrund, nachdem zuvor vor allem funktionale Erweiterungen entwickelt wurden. Dies hatte negative Auswirkungen auf die Nutzer-Zufriedenheit. Hoffen ermittelte 2010, dass mehr als 50% der Saros-Nutzer Saros weniger als fünf Mal verwendet haben, was unter anderem Anstoß für die Masterarbeit von Björn Kahlert [Kahlert2011] war, in der die Out-of-Box-Experience, der Erstkontakt mit dem System, analysiert und verbessert wurde.

Im Anschluss an die Arbeit von Björn Kahlert, erfolgte eine weitere größere Veränderung in der Oberfläche von Saros, die dazu führte, dass die vormals komplett getrennten, einzelnen Eclipse-Views von Saros zusammengeführt wurden, und der Chat, die Liste der Buddies (die damals noch unverständlicherweise „Roster“ hieß) und die Ansicht der Sitzungsteilnehmer in einem einzigen View zusammengefasst wurden.

In der darauf folgenden Bachelorarbeit von Asenij Solovjev, „Evaluation der Mechanismen zum Darstellen der Workspace Awareness in Saros” [Solovjev 2011] wurde evaluiert, welche Mechanismen in Saros zur Übermittlung von Präsenzinformationen vorhanden sind, und welche Informationen nicht übermittelt werden, die bei der nicht-verteilten Paarprogrammierung

grundsätzlich vorliegen (z.B. wohin der andere Programmierer gerade schaut, welche Datei er offen hat). Solovjev prüfte hierbei, in wie weit dies negative Auswirkungen auf die Arbeit mit Saros haben kann. Im Rahmen seiner Bachelorarbeit entstanden Videoaufzeichnungen der Nutzung von Saros (Usability-Tests), sowie Erkenntnisse aus semi-strukturierten Interviews und Fragebögen, die in die Datenbasis dieser Diplomarbeit integriert wurden.

Ergebnis seiner Arbeit war unter anderem, dass der Einladungsprozess, durch mangelnde Rückmeldungen während des Sitzungsaufbaus, als unzusammenhängend empfunden wurde. Auch wurden Mängel bezüglich des Umfangs der übermittelten Workspace Awareness Informationen festgestellt: es fehlen insbesondere Angaben, welcher Nutzer im Follow-Modus ist, wem dieser Nutzer folgt und welche Dateien andere Nutzer geöffnet haben. Die verwendeten Mechanismen zur Darstellung von Präsenzinformationen sind teilweise zu subtil, und teilweise fehlen sie komplett. Diese Erkenntnisse und eigene Auswertungen der Videoaufzeichnungen dienen zur Untermauerung der Relevanz der Usability-Probleme, die in dieser Diplomarbeit identifiziert werden.

Analysen zu Verbesserungsmöglichkeiten, die in diese Arbeit nicht eingebracht wurden, finden sich in der Bachelorarbeit von Alexander Narweleit „Analyse und Evaluation von Verbesserungsmöglichkeiten der User Experience in Saros“ [Narweleit2011]. Er vergleicht Saros mit anderen Software-Systemen, hinsichtlich der Funktionen und verwendeten Konzepte, und analysiert deren Umsetzbarkeit in Saros. Zudem gibt er Empfehlungen, wie die „User Experience“ („Nutzerzufriedenheit“, genaue Begriffsdefinition siehe [Narweleit2011]) verbessert werden könnte. Allerdings konnten die Vorschläge aus der Arbeit, die sich unter anderem auf die Text-Formulierungen in den Dialogen beziehen, keine Zustimmung im Saros-Team finden und wurden damit nicht weiter berücksichtigt.

Da nach der Analyse von Hoffen (2010) und Kahlert (2011) signifikante Veränderungen in der Handhabung von Saros vorgenommen wurden, ist eine erneute Betrachtung der Nutzungsstatistik notwendig, um Rückschlüsse auf die tatsächliche Nutzung des Systems ziehen zu können, die im folgenden Abschnitt beschrieben werden.

2.2 Einblicke in die tatsächlichen Nutzung

Um bei der Analyse der Usability vor allem praxisrelevante Probleme zu ermitteln, ist zunächst zu ermitteln, wie Saros tatsächlich genutzt wird, um dies bei der Evaluation berücksichtigen zu können. Hierzu werden die Daten der in Saros integrierten anonymisierten Nutzungsstatistik ausgewertet.

Diese Statistik liefert Informationen über die Häufigkeit und Verwendungsdauer der grundlegenden Funktionen von Saros, darunter welche Arten von Sitzungen häufig vorkommen, und welche Funktionen innerhalb der Sitzungen genutzt wurden.

Die Nutzungsstatistik kann freiwillig aktiviert und auch wieder deaktiviert werden. Somit gibt es nur Datensätze von Nutzern, die die Nutzungsstatistik aktiviert haben. Über andere Nutzer gibt es keine Informationen. Es liegt nahe zu vermuten, dass Nutzer, die die Nutzungsstatistik abschalten, Saros anders Nutzen, als die, die sie

explizit aktivieren. Insbesondere deshalb, weil Nutzer zum Aktivieren der Statistik im Einrichtungs-Wizard eine letzte, optionale Seite erreichen mussten, oder die Konfigurationsoption in den Saros Einstellungen entdeckt haben müssen. Dies würde auf eine andere Vorgehensweise beim Erkunden der Anwendung hindeuten. Verlässliche Rückschlüsse auf die Eigenschaften der Nutzer können hier aber keine gezogen werden. Da sich bestenfalls vage Vermutungen ergeben würden, wurde auf eine genauere Untersuchung dieser Aspekte verzichtet. Die Daten der Nutzungsstatistik werden als ausreichend repräsentativ angesehen für die Struktur der Analyse und im Folgenden ausgewertet und als Grundlage für die Wahl der Testszenarien für die empirische Evaluation genutzt.

Um verwertbare Daten aus der Nutzungsstatistik mit ca. 25.000 Datensätzen extrahieren zu können, müssen die Daten zunächst von verfälschenden Datensätzen bereinigt werden. Folgende Datensätze wurden entfernt:

- Sitzungen mit DEVEL-Versionen. Dies sind Tests von Saros-Entwicklern, die die Nutzungsstatistik nicht deaktiviert hatten
- Sitzungen mit nur einem Teilnehmer (d.h. es ist niemand der Einladung zur Sitzung gefolgt). Da Saros bereits direkt nach dem Senden einer Einladung eine Sitzung startet, in der sich nur der Host[11] befindet, und die Nutzungsstatistik dies nicht herausfiltert, mussten diese Datensätze ebenfalls entfernt werden
- Sitzungen mit einer Dauer von deutlich mehr als 8h. Es wurde angenommen, dass hier die Nutzer ihre Computer angelassen haben, und die Sitzung über Nacht weiter lief

Insgesamt standen somit nur noch ca. 3800 Datensätze von Saros-Sitzungen zur Verfügung die gefiltert nach Sitzungen aus dem Jahr 2011 folgende Kennzahlen ergaben:

- Die durchschnittliche Sitzungsdauer beträgt 31 Minuten
- Durchschnittlich 1,7 Mal wechselt der der Follow Modus in einer Sitzung
- In ca. 25% der Sitzungen wird mindestens einmal parallel innerhalb derselben Datei editiert
- Der Anteil der sehr langsamen Verbindungsart IBB[12] liegt bei ca. 45% mit einer durchschnittlichen Übertragungsgeschwindigkeit von 13 KB/s
- Die durchschnittliche übertragene Datenmenge betrug (komprimiert) 3.25MB pro Sitzung.

Die durchschnittliche übertragene Datenmenge ist mit 3.25MB recht klein und selbst bei der langsamsten Verbindungsart (IBB) würde die Synchronisation der Daten zum Aufbau einer Sitzung durchschnittlich weniger als 5 Minuten dauern.

Hierbei ist noch zu berücksichtigen, dass die Datenmenge neben der Dateigröße des Projekts auch die spätere Datenübertragung während der laufenden Sitzung beinhaltet, und somit alle Tastaturanschläge und sonstigen Datenübertragungen die stattfanden. So ist nur der Rückschluss möglich, dass die tatsächliche

[11] Der Nutzer der in eine Sitzung einlädt
[12] In-Band-Bytestream, Daten als XML gekapselt, ohne Direktverbindung

durchschnittliche Projektgröße kleiner als die durchschnittliche übertragene Datenmenge ist.

Bei der Verteilung der Betriebssysteme dominiert Windows mit über 70%, Linux (16.57%) und Mac OS X (12.58%) werden eher selten genutzt. Bei der Erwartungskonformität werden somit vor allem Windows-Betriebssysteme eine große Rolle spielen.

Um Hinweise auf realistische Anwendungsszenarien zu gewinnen, wurde betrachtet, welche Rollenverteilung der Teilnehmer in den Sitzungen üblich ist. Bevor die Rollenverteilung in Saros verändert wurde, gab es in Saros „Driver" und „Observer" Rollen, wobei die Observer keine Schreibrechte hatten. Der Großteil (90%) aller Sitzungen bestanden aus 2 Teilnehmern, die einen Rollenwechsel nur am Anfang der Sitzung durchführten, und für den Rest der Sitzung die jeweilige Rolle beibehielten (83% der Sitzungen, Vgl. Auswertung in [Kahlert2011]).

Durch eine Veränderung in der Konzeption von Saros, (unter anderem weil in [Kahlert2011] vermutet wurde, dass Nutzer nicht wissen, dass in einer Sitzung auch mehrere „Driver" mit Schreibrechten möglich sind), wurden die Rollenbezeichnungen entfernt und die Funktionsweise von Saros verändert.

Die neue Konzeption von Saros:

- Der Nutzer, der einlädt hat die Rolle des „Host", die in Saros aber nicht explizit erwähnt wird
- Sitzungen starten mit Schreibrechten für alle Sitzungsteilnehmer
- Nur der Host kann Nutzern Schreibrechte entziehen, und ihnen diese wieder zuweisen
- Eine Übertragung der Host-Rolle ist nicht möglich, somit hat der Host immer Schreibrechte

Der Zeitpunkt der Änderung war nicht genau zu identifizieren, allerdings konnten bei der Auswertung der Statistik noch Datensätze entfernt werden, die die alten Rollenbezeichnungen „Driver" bzw. „Observer" beinhalten. Unklar bleibt, ob die Anpassung der Bezeichnungen in der Statistik zeitgleich mit den Änderungen in der Oberfläche vorgenommen wurde.

Mit den verbleibenden, aktuellen Datensätzen ergibt sich ein Anteil von weniger als 2% von Sitzungen, in denen die initialen Rollenverteilungen verändert wurden. Entweder sehen Nutzer keinen Sinn in einer Änderung der Nutzerrechte, oder die Funktion zum Wechseln der Rechte erschließt sich den Nutzer einfach nicht, da sie nur im Kontextmenü der Nutzer enthalten ist. Auch ist es möglich, dass die Nutzer sich verbal darauf einigen, wer der Driver und wer der Observer ist, und entsprechend auf Schreibaktionen verzichten, ohne explizit die Rechte in der Sitzung zu verändern. Präzise Rückschlüsse sind auf Basis der vorliegenden Daten nicht möglich. Daher wird in Bezug auf die Rechtevergabe vor allem versucht zu ermitteln, ob sich die Funktion den Nutzern von selbst erschließt, oder ob diese nicht genutzt wird, weil sie schlecht zu finden ist.

Es ergeben sich folgende Eigenschaften klassischer Saros-Sitzungen:

- Sitzungen bestehen aus 2 Teilnehmern
- Die Rechtevergabe wird kaum genutzt
- Der Followmodus wird in jeder Sitzung verwendet
- Die VoIP-Funktion wird nicht genutzt (in weniger als 0.5% aller Sitzungen)

Da es nicht ausgeschlossen werden kann, dass Nutzer aufgrund der Instabilität von Saros und mangelnden Vertrauens in die Robustheit des Systems keine Sitzungen mit mehr als zwei Teilnehmern versuchen, werden auch Sitzungen mit drei oder mehr Teilnehmern für die Analyse eingeplant, um zu erkennen ob und welche Probleme es hierbei gibt, die bei Sitzungen mit zwei Teilnehmern nicht auftreten.

Nachdem nun sowohl die gewünschten Anwendungsszenarien als auch die tatsächliche Nutzung von Saros präsentiert wurden, wird im folgenden Kapitel beschrieben, welche Methoden für die Analyse zur Verfügung stehen und diskutiert, inwiefern die Verfahren für die Analyse von Saros geeignet sind.

3 Evaluation der Usability

Die Analyse der Usability von Saros ist ein Kernbestandteil dieser Diplomarbeit. Ziel ist es, zu ermitteln, welche Usability-Probleme in Saros existieren, diese zu katalogisieren und für die Beseitigung zu priorisieren und im Anschluss zu beheben.

In diesem Abschnitt wird zunächst eine Übersicht der verfügbaren Ressourcen präsentiert. Anschließend werden die verfügbaren Analyse-Verfahren vorgestellt und hinsichtlich ihrer Eignung für diese Arbeit diskutiert. Die Erkenntnisse die bei der späteren Anwendung der Verfahren gewonnen wurden, werden im darauffolgenden Kapitel beschrieben.

3.1 Verfügbare Ressourcen

Bei der Eignungsprüfung der Analyseverfahren ist zunächst zusammenzufassen, welche Ressourcen und Datenquellen für diese Arbeit zur Verfügung stehen, die die Wahl der Verfahren deutlich beeinflussen werden.

Für die Analyse kann die aktuelle, frei verfügbare Release-Version von Saros (11.12.9.r3685) verwendet werden, die über die öffentliche Eclipse-Update-Site[13] relativ einfach in einer Eclipse-Instanz installiert werden kann.

Zudem steht durch den Zugriff auf das SVN-Repository[14] der Quellcode des Programms zur Verfügung. Aus Eclipse heraus lässt sich direkt die aktuelle Version von Saros starten, basierend auf dem vorliegenden Programmcode. Somit ist es möglich Prototypen mit experimenteller Implementierung von Nutzern testen zu lassen. Für die Nutzer ergibt sich hierbei kein Unterschied zu der Bedienung der Original–Installation – es muss lediglich im Hintergrund eine Eclipse-Instanz laufen, die den aktuellen Quellcode geladen hat.

Für die Wahl der Testszenarien können neben der bereits beschriebenen Auswertung der Nutzungsstatistik, Daten aus der vorangehenden Arbeit von Björn Kahlert [Kahlert2011], sowie Rückmeldungen von Saros-Nutzern über die Saros-Umfrage verwendet werden. In diesen Fragebögen kommentieren die Anwender auch, was der Sinn und Zweck der durchgeführten Saros-Sitzung war.

Für die Durchführung von Nutzertests konnten bei einem mittelständischen Softwarehersteller, vier Softwareentwickler akquiriert werden, die in zeitlich begrenztem Umfang für Usability-Tests, Interviews und das Ausfüllen von Fragebögen verfügbar waren. Zudem standen andere Mitglieder aus dem Saros-Team für technische Hilfestellungen und Gedankenaustausch zur Verfügung.

Diese Ressourcen bilden die Grundlage für die erste Analyse, und die darauffolgenden Evaluierungen der Veränderungen. Welche Verfahren für die

---

[13] http://dpp.sourceforge.net/update
[14] Freie Versionsverwaltung für Dateien und Verzeichnisse: http://subversion.apache.org/

Analyse von Saros geeignet sind und welche Verfahren nicht verwendet werden konnten wird im folgenden Abschnitt beschrieben.

3.2 Geeignete Analyseverfahren

Evaluierungen können grundsätzlich zur Feststellung eines Zielerreichungsgrades erfolgen, zum Beispiel in Bezug auf den Systemumfang oder die Benutzerzufriedenheit (Vgl. [Sarodnick2006]). So könnte für die Evaluation von Saros zunächst allgemein festgehalten werden, dass die Eignung von Saros zur effizienten, gemeinsamen, räumlich verteilten Programmierung geprüft werden soll.

Da Saros-Releases bereits mehrere Jahre lang öffentlich verfügbar sind, und produktiv eingesetzt werden, wie sich aus den Nutzer-Umfragen ergibt, erübrigen sich Verfahren, die darauf abzielen, die Usability von Systemen zu verbessern, die gerade neu entwickelt werden, bzw. sich allgemein in einer frühen Entwicklungsphase befinden. Es war nicht das Ziel, und es wäre auch zeitlich unmöglich gewesen, Saros komplett neu zu gestalten.

Grundsätzlich wird bei Evaluationen zwischen formativen und summativen Evaluationen unterschieden. Eine summative Evaluation erfolgt nach Abschluss eines Entwicklungsprozesses, und dient zum Beispiel dazu, ein Produkt mit anderen Produkten zu vergleichen, und um zu prüfen, ob es durch die erfolgten Entwicklungsarbeiten tatsächlich Verbesserungen gab. Hierbei kommen meist quantitative Verfahren zum Einsatz (Vgl. [Sarodnick2006]). Bei einer formativen Evaluation bedient man sich vor allem qualitativer Verfahren, darunter Usability Tests, unter Einsatz der Thinking Aloud Methode. Eine formative Evaluation ist ein wichtiger Bestandteil des Entwicklungsprozesses, und führt durch einen Zyklus aus Design und Evaluation von Prototypen zu guten Ergebnissen (Vgl. [Holz1993], [Nielsen1993]). Da es nicht das Ziel war, lediglich einen Abschlussbericht über den Zustand der Usability von Saros zu liefern, sondern konstruktiv am System zu arbeiten, liegt der Fokus auf einer formativen Evaluation.

Zum Erstellen einer Übersicht der verfügbaren Verfahren für die Usability-Analyse, bietet es sich an, diese zunächst zu klassifizieren. Whitefield et al. [Whitefield1991] wählen hierfür folgende zwei Dimensionen:

1. Evaluationen mit echten Nutzern, oder ohne Nutzer zu involvieren
2. Evaluation einer fertig implementierten Benutzungsoberfläche oder Tests von Prototypen

Alle genannten Varianten werden in dieser Arbeit genutzt. So werden theorieorientierte Evaluationen, darunter Reviews mit Hilfe von Normen und Styleguides/Richtlinien, ohne Nutzerbeteiligung durchgeführt. Anschließend folgen benutzerorientierte Evaluationen, mit den vier Probanden, die für Interviews, Fragebögen, Beobachtungen, Erfahrungsberichte, sogenannte „Think-Aloud-Protokolle“ und schließlich kontrollierte Experimente zur Verfügung stehen (Vgl. [Herczeg2005]).

Die von Herczeg (2005) aufgeführten Modellierungsverfahren, aus der Gruppe der aufgabenorientierten Evaluation, wie z.B. der GOMS-Modellierung kommen nicht

zum Einsatz. Sie werden aufgrund der hohen Komplexität und Formalität als ungeeignet für die Analyse angesehen.

Die für die Analyse grundsätzlich geeigneten Verfahren sind somit die folgenden:

Theorieorientierte analytische Evaluation:

- Heuristische Evaluation auf Basis von etablierten Richtlinien
- Cognitive Walkthrough
- Pluralistic Walkthrough (mit einer Gruppe von Evaluatoren)

Empirische Evaluation mit Nutzern:

- Durchführung von Usability-Tests
    - Thinking aloud Protokolle
    - Freie Beobachtung von Nutzern
- Fragebögen & Interviews
- Nutzer-Feedback durch Fragebögen (fortlaufend)
- Nutzungs-Protokolle des Systems (fortlaufend)

Die analytischen Verfahren sind stark beeinflusst von den Erfahrungen der Usability-Experten, die die Analyse als Gutachter durchführen. Um dem entgegen zu wirken empfiehlt Nielsen (1993) eine Kombination aus unterschiedlichen Methoden zu wählen, insbesondere da empirische Methoden dazu beitragen würden, die praktische Relevanz der gefundenen Probleme, die nach den analytischen Evaluationen nur in Form von Abweichungen von Richtlinien und Standards vorliegen, zu ermitteln (Vgl. [Sarodnick2006] und [Nielsen1993]). Daher werden in dieser Arbeit mehrere Verfahren kombiniert.

Bei der folgenden Betrachtung der Vor- und Nachteile der Verfahren und der Anforderungen an die Ressourcen, zur Anwendung der Verfahren, wird in dieser Arbeit vor allem auf die Analyseergebnisse von Nielsen ([Nielsen1993], [Nielsen2000]) zurückgegriffen. Im Anhang 7.1 findet sich eine tabellarische Übersicht der Verfahren und deren Anforderungen. Diese beinhaltet eine Angabe der größten Vorteile und Nachteile der Verfahren, sowie die Anforderungen an die Menge an Nutzern, die für die Durchführung erforderlich sind, die bei der Diskussion der Eignung der Verfahren berücksichtigt wurden.

3.3 Heuristische Evaluation

In der heuristischen Evaluation vergleichen Evaluatoren (idealerweise Usability-Experten) die Software mit gängigen Usability-Richtlinien, sogenannten Heuristiken. Die Experten versuchen hierbei die Rolle eines späteren Nutzers einzunehmen, und bei der Nutzung des Systems realistische Szenarien zu betrachten. Abweichungen von diesen Heuristiken deuten meist auf ein Usability-Problem hin, das dann in Usability-Tests bestätigt werden sollte (Vgl. [Nielsen1993], [Sarodnick2006]).

Der größte Vorteil für die heuristische Evaluation ist nach Nielsen (1993), dass individuelle Usability-Probleme gefunden werden können, die vom Evaluator abhängen. Das größte Risiko liege hierbei darin, dass keine „echten Nutzer" involviert seien, und somit keine „Überraschungen" durch besondere Bedürfnisse des Nutzers auftreten [Nielsen1993, S.224].

Bei der Prüfung können umfangreiche Listen von Heuristiken zum Einsatz kommen, die bis zu tausend Heuristiken umfassen (z.B. [Smith1986] mit 944 Richtlinien). Entsprechend aufwändig gestaltet sich eine solche Analyse. Allerdings genügt es nach Nielsen (1993) meist eine Analyse auf Grundlage von einigen wenigen, allgemeinen Heuristiken durchzuführen, die in Ihrer Beschreibung weniger Präzise sind, aber grundsätzlich alle Arten von Usability-Problemen erfassen können. Eine erste Liste von allgemeinen Heuristiken haben Nielsen und Molich [NM90] entwickelt, die dann kontinuierlich überarbeitet wurde [Nielsen1994]. Derartige allgemeine Listen von Heuristiken, werden von Sarodnick und Brau (2006) dann als kategorisierungsfreundlich bezeichnet, wenn eine eindeutige Zuordnung eines Problems zu einer Heuristik erfolgen kann. Daher wurde die Liste um weitere Heuristiken wie „Joy of use", und „interkulturelle Aspekte" ergänzt, und enthält in einer aktuellen Fassung 12 Heuristiken [Sarodnick2006].

Identifizierte Abweichungen von den verwendeten Heuristiken, werden durch die Evaluatoren mit einer Gewichtung versehen, die angibt, wie schwerwiegend das Problem ist (und ob es überhaupt als ein Usability-Problem anzusehen ist, siehe Abschnitt 3.9 Verfahren zur Katalogisierung der Probleme). Erst nach Abschluss der Evaluation durch alle Evaluatoren werden die Problemkataloge zusammengeführt und die Gewichtungen der einzelnen Evaluatoren gemittelt, um eine verlässlichere Gewichtung zu erhalten. Die Gewichtung und die Analyse erfolgt hierbei zunächst getrennt, damit die Evaluatoren das System unabhängig voneinander und unvoreingenommen betrachten [Nielsen1993, S.157]. Der Prüfung der Oberfläche des zu testenden Systems, sollte bei der Evaluation durch die Gutachter mehrfach durchgeführt werden. Zunächst sollte eine grundlegende Kenntnis des Systems erlangt werden, bevor in einem zweiten Durchgang alle Punkte der Oberfläche auf Abweichungen von Heuristiken geprüft werden sollen. Mindestens zwei Durchläufe sind zu empfehlen, auch wenn es letztlich eine Entscheidung des Evaluators ist, wie viele Durchläufe insgesamt durchgeführt werden [Nielsen1993, S.158].

Die heuristische Evaluation liefert schließlich eine Liste von Usability Problemen, wobei Nielsen (1993) kritisiert, dass dabei kein systematischer Weg zur Behebung der Probleme angeboten wird. Allerdings sei es leicht, durch die Referenz zu der

verletzten Heuristik, ein neues Konzept zu entwerfen, dass die verletzten Heuristiken nicht verletzt. Zudem hätten viele identifizierte Usability-Probleme offensichtliche Lösungen, sobald das Problem als solches erkannt wurde. Weitere Verbesserungen der Effektivität der Evaluation können nach Dykstra (1993) auch durch eine Erweiterung der Heuristiken, um domänenspezifische Themen erreicht werden (Vgl. auch [Sarodnick2006], [Nielsen1993]). Hierfür seien allerdings Erfahrungen aus Usability-Tests mit vergleichbaren Produkten nötig, die nicht vorliegen.

Da Saros bereits als fertige Software vorliegt, kann somit mit der heuristischen Analyse geprüft werden, da diese nach Sarodnick und Brau (2006) und Nielsen (1993) auch für die Evaluation von Prototypen (z.B. Papier-Prototypen) geeignet wäre. Es besteht die Möglichkeit, dass bereits bei der heuristischen Analyse kostengünstig viele Usability-Probleme identifiziert werden können, ohne hierfür Nutzer involvieren zu müssen. Da die Evaluation im Rahmen dieser Diplomarbeit alleine durchgeführt wird, entfällt die sonst übliche Mittelung der Problem-Gewichtungen, weshalb persönliche Präferenzen bei der Problemgewichtung ein Problem darstellen werden (Vgl. [Nielsen1993]). Um dem entgegen zu wirken, werden die identifizierten Probleme mit dem Saros-Team diskutiert, um eine bessere Einschätzung der Problemschwere zu gewinnen.

Bei der Analyse werden nicht die 12 Heuristiken von Sarodnick und Brau verwendet, da es sich im ersten von zwei Durchläufen der heuristischen Evaluation als schwierig erwies, die identifizierten Probleme den Heuristiken zuzuordnen. Den neu hinzugefügten Heuristiken „Joy of use“ und „interkulturelle Aspekte“ konnten keine der Probleme zugewiesen werden. Da der Problemkatalog als Grundlage für weitere Verbesserungen von Saros dienen soll, wird in dieser Arbeit auf die Liste mit 10 Heuristiken von Nielsen (1994) zurückgegriffen. Diese sind für Entwickler, aus eigener Erfahrung heraus, leichter zu verstehen und die Beschreibungen der Heuristiken sind eher als eine Handlungsempfehlung zu verwenden.

Die allgemeinen Usability Heuristiken von Jakob Nielsen, frei übersetzt aus [Nielsen1994]:

N1. Sichtbarkeit des Systemstatus
Der Nutzer sollte vom System durch geeignetes Feedback in angemessener Zeit immer darüber informiert werden, was er gerade macht und wo er sich befindet.

N2. Übereinstimmung zwischen System und realer Welt
Das System sollte die Sprache des Benutzers sprechen. Dafür soll es keine systemorientierten Begriffe, sondern Wörter und Formulierungen, die dem Nutzer geläufig sind, nutzen.

N3. Benutzerkontrolle und Freiheit
Es muss immer ein „Notausgang“ für den Nutzer vorhanden sein, den er ohne große Probleme betätigen kann, z.B. wenn er unabsichtlich eine Aktion ausgelöst hat. Dafür sollten auch die Funktionen „Undo“ und „Redo“ unterstützt werden.

N4. Konsistenz und Standards
Plattform-Konventionen müssen eingehalten werden. Den Nutzern muss klar sein, welche Begriffe die gleiche Bedeutung haben und welche Aktionen dasselbe Ziel verfolgen.

N5. Fehler vermeiden
Das System soll so ausgelegt sein, dass Fehler bereits im Vorhinein vermieden werden.

N6. Erkennen vor Erinnern
Objekte, Aktionen und Optionen sollten für den Nutzer immer sichtbar sein. Dadurch bleibt ihm erspart, sich bereits gesehene Informationen oder Abläufe zu merken.

N7. Flexibilität und effiziente Nutzung
Es sollte Nutzern ermöglicht werden, von ihnen häufig durchgeführte Aktionen auf ihre Bedürfnisse anzupassen. Um das System für Anfänger und Experten besser bedienbar zu machen, kann das System zum Beispiel Beschleunigungsmöglichkeiten bereitstellen, die nur der erfahrene Nutzer erkennt und somit die eigene Interaktionsgeschwindigkeit steigern kann.

N8. Ästhetisches und minimalistisches Design
Irrelevante Informationen sollten möglichst vermieden werden, da hierdurch die Sichtbarkeit von relevanten Informationen gefährdet sein kann.

N9. Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern
Bereitgestellte Fehlermeldungen müssen für den Benutzer auch verständlich sein. Das Problem sollte deshalb in der Sprache des Nutzers beschrieben und mögliche Lösungsvorschläge aufgezeigt werden.

N10. Hilfe und Dokumentation
Jedem Nutzer sollte der Zugang zu einer Dokumentation und Hilfe möglich sein. Die Hilfe sollte dabei leicht zu durchsuchen sein und die für den Nutzer relevanten Aufgaben und konkreten Schritte beinhalten.

Nach Nielsen (1993) sind insbesondere die Anzahl der Evaluatoren und deren Domänenexpertise entscheidend für die Qualität der Ergebnisse. Nach einer Studie über sechs Projekte, fand ein einzelner Evaluator nur 35% der Usability Probleme [Nielsen1993, S.156]. Ein einzelner Evaluator wird die meisten Usability-Probleme in einer Oberfläche übersehen, weshalb mindestens der Einsatz von 3, besser aber 5 Evaluatoren empfohlen wird.

Aufgrund meiner Tätigkeit als Softwareentwickler in verteilten Teams, entspreche ich der Zielgruppe von Saros und verfüge über Domänenwissen im Bereich der Softwareentwicklung. Ich bin ein „echter Nutzer“ und habe Saros bereits produktiv eingesetzt, weshalb ich hier keine Probleme erwarte. Da allerdings die Evaluation alleine erfolgen wird, muss die Anzahl der identifizierten Probleme über eine Kombination mit weiteren Verfahren vergrößert werden (Vgl. [Sarodnick2006], [Nielsen1993]). Auch sollte die Relevanz der gefundenen Probleme, durch Nutzertests und Erkenntnisse aus den Nutzerumfragen, untermauert werden, weshalb die Heuristische Evaluation im Anschluss durch Usability Tests ergänzt wird.

Da es bei der Evaluation vorkommen kann, dass eine einzelne Komponente, z.B. ein Wizard, gleich gegen mehrere Heuristiken verstößt, sollte hierfür vorab ein entsprechendes Konzept für die Katalogisierung entwickelt werden. Die Fehler könnten separat aufgelistet oder zusammengefasst werden. Bei einer zusammengefassten Dokumentation bestünde das Risiko, dass einzeln aufgetretene Fehler an anderer Stelle wiederholt werden, da diese von den Entwicklern nicht einzeln betrachtet werden. Auch wäre nicht sichergestellt, dass alle Probleme eines Oberflächenelements vollständig behoben werden können, sodass ein Abarbeiten des Problemkatalogs sich schwieriger gestaltet (Vgl. [Dykstra1993]).

Nach Dykstra (1993) sollten Probleme daher solange zerlegt werden, bis sie genau einer Heuristik zugeordnet werden können, da eine Zuweisung zu mehreren Heuristiken auf ein zu allgemein beschriebenes Problem hinweist. Beim Aufbau des Problemkatalogs wurde allerdings festgestellt, dass dieser schnell eine Dimension annahm, die auf Entwickler „erschlagend" wirken kann. Bereits bei der Wahl der Heuristiken raten Nielsen (1993) und Sarodnick und Brau (2006) davon ab, zu große Listen zu verwenden, da diese abschreckend wirken können.

In Anbetracht der Tatsache, dass einige Komponenten gegen fast alle Heuristiken verstoßen, stellte sich die Frage, ob der hohe Mehraufwand noch zu rechtfertigen wäre, da die betroffene Komponente von Grund auf neu gestaltet werden muss. Da kein Mehrwert in der detaillierten, getrennten Dokumentation der einzelnen Verstöße gesehen wurde, finden sich im Katalog einige Einträge, die Verstöße gegen mehrere Heuristiken in einer Komponente beschreiben.

3.3.1 Eclipse Guideline / Gestaltungsrichtlinien

Neben den bereits vorgestellten Heuristiken, gibt es noch Gestaltungsrichtlinien (engl. Guidelines), die meist mit grafischen Beispielen und der Beschreibung von Design-Prinzipien aufwarten, die genau wie die Heuristiken, empirisch erprobte und von Usability Experten anerkannte Empfehlungen darstellen.

Da Saros ein Eclipse-Plugin ist, ist es wichtig, dass das Look & Feel der Applikation möglichst nahe an der in Eclipse üblichen Gestaltung orientiert ist (Vgl. Heuristik H5. Erwartungskonformität [Sarodnick2006] oder N4. Konsistenz und Standards [Nielsen1993]). Konsistenz und Standards spielen besonders bei der Lernförderlichkeit eine große Rolle (Vgl. [Dix2003]). Da einige Funktionen von Saros der Statistik zufolge kaum genutzt werden, soll die Integration von Saros in Eclipse und die Erwartungskonformität allgemein überprüft werden.

Für Eclipse steht dazu, neben den Top 10 UI Guidelines[15] auf der Webseite, auch eine umfangreichere Beschreibung der „User Interface Guideline"[16] bereit. In dieser wird detailliert beschrieben, wie Wizards in Eclipse gestaltet sein sollten.

Diese Guidelines werden als eine weitere Grundlage für die Prüfung der Usability von Saros verwendet. Da Saros primär von Windows-Nutzern eingesetzt wird, werden an einigen Stellen auch die Microsoft Guidelines herangezogen, so z.B. bei der Umgestaltung der Kontextmenüs von Saros. Neben der Verwendung der Guidelines zur Prüfung auf Abweichungen, dienen diese auch als Grundlage für die spätere Neugestaltung der Saros-Oberfläche.

3.4 Cognitive Walkthrough

Bei einem "Cognitive Walkthrough" wird die Evaluation auf die Aufgaben der Nutzer, die Oberfläche und auf die Erlernbarkeit des Systems fokussiert. Nach einer Identifikation der Zielgruppe und der Ermittlung von realistischen Beispiel-

[15] http://wiki.eclipse.org/Top_Ten_Lists_Working_Page
[16] http://wiki.eclipse.org/User_Interface_Guidelines

Aufgaben für die Evaluation, werden Szenarios definiert, mit Handlungsabfolgen zur Bedienung des Systems.

Evaluatoren versuchen dann die vordefinierten Aufgaben mit der gegebenen Bedienoberfläche zu lösen und notieren dabei Probleme im Handlungsablauf (und Lösungsvorschläge). Zudem werden Annahmen über die Benutzererfahrung vorgenommen. Welches Ziel verfolgt der Nutzer bei Bedienung einer Komponente, und wieso ist dies sein Ziel. Welche Aktionen sind offensichtlich in der Oberfläche verfügbar und sind diese so benannt, dass sie eindeutig der Zielerfüllung des Nutzers zuzuordnen sind. Auch wird erfasst, ob Nutzer bei der Wahl der richtigen Aktion, geeignete Rückmeldung erhalten, oder versuchen werden, ihre Aktion rückgängig zu machen oder zu wiederholen [Wharton1994].

Diese Methode kann auch mit Prototypen angewandt werden und wäre damit auch für die Analyse der Lösungsansätze in der zweiten Phase der Arbeit geeignet.

Da bereits eine Heuristische Evaluation durchgeführt wird, bei der eine intuitive Herangehensweise darin besteht, zunächst zu überlegen, welche Ziele ein Nutzer bei der Bedienung der Oberflächenkomponenten erzielen möchte (und wofür diese gedacht waren) wird auf das explizite Anwenden dieses Verfahrens verzichtet. Die Ideen aus dem Konzept des Cognitive Walkthrough fließen bereits in die Durchführung der Heuristischen Evaluation mit ein.

3.5 Pluralistic Walkthrough

Bei einem „Pluralistic Walkthrough“ wird eine heuristische Analyse durch repräsentative Nutzer, Produkt-Entwickler, Designer und Usability-Experten durchgeführt. Die Nutzer bringen hierbei Ihre Domänenexpertise ein, so dass das Verfahren besonders geeignet für den Fall ist, dass nur „single expert“ Usability Spezialisten zur Verfügung stehen, die kein Domänenwissen aufweisen können [Nielsen1993, S.162].

Da ich sowohl Usability- als auch Domänenexperte bin, fällt damit einer der großen Vorteile dieses Verfahrens weg. Aufgrund der hohen Anforderungen an die Anzahl der Teilnehmer entfällt diese Methode ebenfalls.

Eine Kombination aus Heuristischer Evaluation und Usability-Tests verspricht bereits eine ausreichende Qualität der Ergebnisse (Vgl. [Nielsen1993]), sodass hier auf weitere aufwändige Verfahren verzichtet wird.

3.6 Usability Tests

Usability-Tests sind eine der besten Methoden um Usability-Probleme in Software zu finden und die Ergebnisse aus einer analytischen Evaluation zu bestätigen und präzisieren. Usability-Tests mit Nutzer liefern direkt Informationen darüber, wie Nutzer tatsächlich mit dem System interagieren und welche Probleme durch unerwartetes Nutzerverhalten auftreten [Nielsen1993, S.165]

Hierbei werden Probanden bei der Nutzung des Systems, bzw. der zu vergleichenden Systeme durch Usability-Experten beobachtet. Zunächst wird

meist ein Usability-Labor vorbereitet, in dem vorkonfigurierte Systeme für die Testdurchführung bereitstehen, sowie Technik für die Aufzeichnung und Überwachung der Testdurchführung. Die Beobachter der Probanden können sich dabei sowohl im selben Raum mit diesen befinden, als auch in einem separaten Raum, der durch einseitig verspiegeltes Glas abgetrennt ist, um Beobachter und Probanden voneinander zu trennen. Bei Problemen mit der Lösung der Aufgabenstellung durch Bedienschwierigkeiten greifen die Beobachter grundsätzlich nur ein, wenn der Proband stark frustriert ist und explizit um Hilfe bittet, da er oder sie sich nicht im Stande sieht die Aufgabe zu lösen. Auch kann eingegriffen werden, wenn beispielsweise technische Probleme aufgetreten sind, die eine weitere Bearbeitung der Aufgabenstellung verhindern würden und für die der Nutzer keine Lösung findet (Vgl. [Nielsen1993]).

Die Beobachter fertigen während der Usability-Tests ein Protokoll an, in dem sie festhalten, wann und wie Probanden Schwierigkeiten mit der Bedienung, oder mit dem Verständnis des Systems, hatten. Sie notieren ebenfalls, ob und wie die Probanden eigenständig eine Lösung finden konnten. Auch die Dauer der Aufgabenbearbeitung kann erfasst werden, um bei weiteren Usability-Tests des verbesserten Systems, oder in Bezug auf ein alternatives System, zu vergleichen, ob diese verbessert werden konnte.

Entscheidend für die Qualität der Ergebnisse von Usability Tests, in Bezug auf Reliabilität und Validität, sind neben der Wahl der richtigen Probanden auch die Aufgabenstellungen, die Vorbereitung der Technik, sowie die Wahl der Evaluatoren. Diese Anforderungen werden im Folgenden beschrieben.

3.6.1 Notwendige Anzahl der Probanden

Die optimale Anzahl an Probanden für Nutzertests wird meist bezogen auf ein Kosten/Nutzen-Verhältnis angegeben. Mehr Probanden zu organisieren, verursacht höhere Kosten (Zeitaufwand für Planung, ggf. Honorar für die Probanden) und vergrößert den zeitlichen Aufwand der Koordination der Tests, sowie der Auswertung der Aufzeichnungen. Mehr Probanden einzusetzen bringt allerdings nicht zwangsläufig mehr und bessere Ergebnisse. [Nielsen2000]

Jakob Nielsen beschreibt in seinem Artikel „Why you only need to test with 5 users“ [Nielsen2000] die Forschungsergebnisse der vorhergehenden Jahre: Ab einer Anzahl von mehr als 5 Probanden steigt die Wahrscheinlichkeit stark an, das man immer wieder dieselben Probleme beobachtet, die man bereits bei anderen Nutzern entdeckt hat. Um (theoretisch) 100% aller Usability-Probleme zu ermitteln, wären zwar mindestens 15 Probanden nötig, allerdings argumentiert er weiter, dass das Ziel von Usability Engineering darin liegt, nicht nur die Usability Probleme zu ermitteln und zu dokumentieren, sondern diese auch zu beheben. Dies deckt sich mit dem Ziel der Diplomarbeit, in der neben einer Dokumentation der vorhandenen Probleme, auch eine Verbesserung der Usability erreicht werden soll.

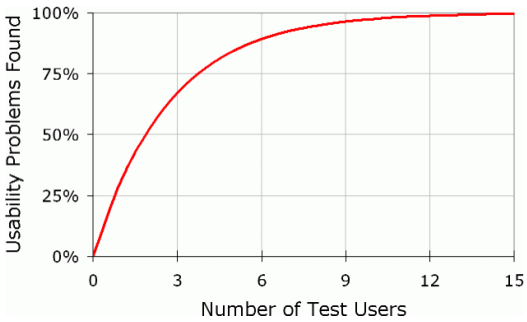

Abbildung 5: Verhältnis zwischen Anzahl der Nutzer und Prozentsatz der gefundenen Probleme, Jakob Nielsen's Alertbox, 19.03.2000 (Quelle: http://www.useit.com/alertbox/20000319.html)

Die Grafik zeigt, dass bis zu einer Anzahl von 5 Nutzern der prozentuale Anteil an gefundenen Usability-Problemen signifikant ansteigt, und weitere Nutzer dann nur noch einen marginalen Vorteil bieten. Das Optimale Kosten-Nutzen-Verhältnis liegt bei 3-5 Nutzern.

Anstatt alle verfügbaren Probanden (z.B. bei 15) für einen großen Usability-Tests zu nutzen, empfiehlt Nielsen eine Reihe kleinerer Usability-Tests mit 3-5 Probanden (Vgl. [Nielsen1993]).

Nach den ersten Tests sollten durchschnittlich 85% der Usability Probleme identifiziert worden sein, die zunächst gewichtet, für die Behebung priorisiert und dann bearbeitet werden sollten (Vgl. [Nielsen2000]). Das größte Problem bei der Behebung dieser gefundenen Usability Probleme sei dabei, dass nicht sichergestellt ist, dass das Lösungskonzept und die Umsetzung des Konzepts die Probleme tatsächlich beheben. Es besteht die Möglichkeit, dass zwar das betrachtete Usability-Problem behoben wurde, aber durch die Änderungen neue Usability-Probleme entstanden sind, oder zuvor verdeckte, aber bereits existierende Usability-Probleme erstmalig zum Vorschein kommen [Nielsen2000]. Daher sollten im Anschluss weitere Testläufe nach der Neugestaltung offenbaren, ob die Probleme zur Zufriedenstellung gelöst wurden. Hierbei sollten die verbleibenden 15% der Original-Usability-Probleme durch die Nutzer gefunden werden, die bei dem ersten Test nicht identifiziert wurden.

3.6.2 Aufgaben für die Probanden

Bei der Durchführung der Testläufe ist die Wahl der Aufgaben entscheidend für die Qualität der Ergebnisse. Aufgabenstellungen sollten weder frivol, noch lustig oder anstößig sein, sondern möglichst nah an realistischer Nutzung des Systems orientiert. Als Quelle für mögliche Aufgabenstellungen dienen Protokolle der Nutzungsfrequenz von Systemfunktionen, sowie Ergebnisse aus Feldstudien und Befragungen von tatsächlichen Nutzern des Systems (Vgl. [Nielsen1993, S. 185f], [Sarodnick2006]), die für diese Arbeit vorliegen.

Die Aufgaben sollen dabei so gestaltet sein, dass eine Erfüllung der Aufgabe innerhalb der beschränkten Zeit für den Usability-Test realistisch ist, ohne dabei trivial zu sein. Sie sollten den Probanden schriftlich zur Verfügung gestellt werden, und dabei deutlich dokumentieren, welches Ergebnis bei der Bearbeitung der Aufgabe erzielt werden soll. Die Probanden sollten zudem die Möglichkeit

bekommen, zunächst Fragen zur Aufgabenstellung zu stellen, um Missverständnisse zu vermeiden. Die erste Aufgabe sollte einfach sein, sodass die Probanden ein erstes Erfolgserlebnis haben, was die Moral verbessert.

Da die Nutzer auch Menschen mit Gefühlen sind, muss bei der Durchführung der Test berücksichtigt werden, dass Nutzer unvermeidbar Fehler machen werden, und das System nur langsam erlernen, wodurch sie ihre eigene Leistung als ungenügend empfinden können, oder sich sogar dumm vorkommen, und dadurch negativen, emotionalen Stress erfahren. Es ist daher unabdingbar, dass den Nutzern vor dem Test klar gemacht wird, dass das System getestet wird, und nicht die Nutzer selbst (Vgl. [Sarodnick2006], [Nielsen1993]).

3.6.3 Eignung für die Analyse von Saros

Das Verfahren erscheint grundsätzlich geeignet für die Analyse von Saros und hatte sich bereits bei der Analyse der Out-of-Box-Experience durch Björn Kahlert bewährt.

Da neben den vier Probanden, die nach Nielsens Studien bereits ausreichend sind für Nutzertests, die Probleme von drei weiteren Nutzern, in den Videoaufzeichnungen aus der Arbeit von Arsenij Solovjev [Solovjev2011], beobachtet werden können, können insgesamt Daten von sieben Probanden ausgewertet werden. Die Anforderungen an die Probanden werden von diesen erfüllt, da sie bereit sind, sich zunächst in Saros einzuarbeiten und schon erste Erfahrungen mit der Paarprogrammierung, und verteiltem Arbeiten haben (über Screensharing Lösungen wie Teamviewer). Auch die notwendigen Kenntnisse der Entwicklungsumgebung Eclipse sind gegeben.

Da für diese Arbeit gewünscht war, dass Nutzer gefunden werden, die bereit sind, bei ihrer regulären Arbeit, z.B. innerhalb einer Softwarefirma, ihre üblichen Arbeitsaufgaben unter Einsatz von Saros zu bewältigen, muss in dieser Arbeit allerdings eher eine Art „Feldbeobachtung“, als ein „Labortest“ durchgeführt werden, da die Entwickler keine Zeit für Tests in einem externen Usability-Labor haben. Die Probanden werden in Ihrem normalen Arbeitsumfeld im Büro beobachtet, wobei ich erwarte, auch Probleme im Zusammenhang mit der vorhandenen Software und Hardware identifizieren zu können.

Bei den Tests wird den Probanden wird üblicherweise eine Liste von Aufgaben präsentiert, die sie mit dem zu testenden System lösen sollen. Bei der Bearbeitung der Aufgaben sollen die Probanden stets laut kommentieren, welchen Gedankengängen sie gerade folgen und wie sie die Rückmeldungen und das Verhalten des Systems interpretieren. Dies ermöglicht es den Beobachtern nachzuvollziehen, wie schwer es für die Nutzer ist, bestimmte Funktionen des Systems zu entdecken, zu verstehen und zu verwenden. Insbesondere können so Abweichungen des mentalen Modells des Nutzers erkannt werden. Dieser Teil des Verfahrens nennt sich „Thinking Aloud“ (Vgl. [Sarodnick2006], [Nielsen1993, S.195].

Hier sind zwei Aspekte problematisch: Erstens sind die Aufgaben zum Teil vorgegeben, da die Probanden ihre normale Arbeit erledigen wollen. Zwar konnten auch für mehrere Szenarien feste Aufgaben vorgegeben werden, allerdings besteht ein Teil der durchgeführten Usability-Tests auf Beobachtungen

der Nutzer bei der Bearbeitung von Aufgaben, die diese selbst gewählt haben (oder aus dem Unternehmen vorgegeben bekamen).

Zweitens besteht laut Nielsen (1993) ein großes Problem beim Thinking Aloud Verfahren vor allem darin, dass es für die Nutzer unnatürlich wirkt, was nicht nur das Testen an sich erschwert – sondern auch die Ergebnisse beeinflussen kann. So verlangsamt das laute Aussprechen die Nutzer möglicherweise bei der Aufgabenbearbeitung, wodurch Performance-Messungen weniger repräsentativ für die tatsächliche Arbeitsgeschwindigkeit der Probanden sind. Auch kann das laute Aussprechen den Denkprozess der Probanden beeinflussen. Sie können dadurch die Bedienung einer Oberfläche schneller oder anders erlernen, als dies sonst der Fall wäre [Nielsen1993, S. 196].

Bei der Analyse von Saros erscheint der Einsatz von Thinking Aloud zunächst als ungeeignet, da die Probanden eine echte Arbeitsaufgabe erledigen sollen. Thinking Aloud könnte sie dabei behindern. Auch ist Saros vor allem ein Tool für die gemeinsame Arbeit an Softwareprojekten. Die Tests werden zwangsläufig mit 2 oder mehr Nutzern durchgeführt, da die Analyse sich vor allem mit der Usability bei Code-Reviews und Paarprogrammierung beschäftigt. Bei diesen Anwendungsfällen erfolgt eine intensive Kommunikation zwischen den Nutzern zur Bewältigung der Aufgabe – lautes Aussprechen der Gedanken über die Nutzung des Tools könnte hierbei hinderlich sein. Sarodnick und Brau (2006) beschreiben, dass eine konstante Aussprache der Gedanken ungeübten Nutzer schwer fällt, dieses Problem aber bei Gruppenarbeit gemindert wird. Tatsächlich ergibt sich durch die Gruppenarbeit in Saros ein Vorteil, da hier ein abgewandeltes Verfahren zum Einsatz kommen kann, dass im folgenden Abschnitt beschrieben wird.

3.6.4 Codiscovery Learning

Es gibt ein alternatives Verfahren zum Thinking Aloud, das sogenannte „Constructive Interaction“, bei dem zwei Nutzer ein System zusammen verwenden und dabei gemeinsam erkunden. Dieses Verfahren wird auch als „Codiscovery Learning“ bezeichnet. Bei diesem Verfahren erarbeiten zwei Nutzer gemeinsam Lösungsansätze zur Bearbeitung der Aufgabenstellung. Vorteil dieses Verfahrens ist dabei, dass die Nutzer bereits daran gewöhnt sind, bei gemeinsamer Problemlösung ihre Gedanken auszusprechen. So ergeben sich tendenziell mehr Kommentare durch die Nutzer, als wenn sie nur für den Beobachter des Tests laut aussprechen würden (Vgl. [Nielsen1993, S.198], [Als2005], [Haak2004]). Nach Haak (2004) sei ein weiterer Vorteil, dass Constructive Interaction von den Probanden positiver empfunden wird, als das klassische Thinking Aloud. Dies könnte unter anderem daran liegen, dass die Probanden sich kontinuierlich über die Oberfläche austauschen können, und durch den Partner eine ständige Hilfestellung existiert.

Dieser Ansatz wird daher für die Durchführung der Usability Tests beim Einsatz von Saros verwendet, da die Probanden ohnehin miteinander kommunizieren werden, um die Aufgaben zu lösen. Auch wird hier insbesondere erwartet, dass das gegenseitige Erklären der Interpretation des Systems durch die Nutzer, Einblicke in deren mentales Modell gewährt, und somit mögliche Verständnisschwierigkeiten offenbart, die sonst schwer zu identifizieren wären.

Zwar besteht die Möglichkeit, dass die Nutzer unterschiedliche Lernstrategien verfolgen und wechselweise unterschiedliche Arten der Bedienung des Systems angewandt werden. Auch kann es sein, dass die beiden Nutzer schlicht nicht miteinander arbeiten können [Nielsen1994, S.198]. Da die Nutzer aber bereits vor den Usability-Tests im Unternehmen zusammen gearbeitet haben, werden hier keine Probleme erwartet.

3.6.5 Identifikation der Probleme als Beobachter eines Usability-Tests

Der wichtigste Teil der Usability-Tests ist die Identifikation der Usability-Probleme. Um als Beobachter eines Usability-Tests zu erkennen, dass gerade ein Usability-Problem auftritt, ist ein Leitfaden für die Beobachtung notwendig. Da die Ergebnisse der Usability-Tests stark abhängig sind von den Beobachtern (Vgl. [Sarodnick2006], [Nielsen1993]), entwickelten Saadawi et Al. (2005) ein Tool zur automatisierten Ermittlung von Usability Problemen aus Think-Aloud-Protokollen. Hier werden, zur Anwendung von Standard-Kriterien für die Auswertung der Protokolle, zu bearbeitende Aufgaben zunächst in einzelne Teilaufgaben unterteilt. Die Identifikationsmerkmale für ein Usability Problem sind hierbei:

1. Die Aufgabe konnte nicht erfolgreich gelöst werden
2. Der Proband braucht mehr als 2 Minuten für die Fertigstellung einer Aufgabenstellung
3. Der Nutzer brauchte mehrere Versuche bis er ein Teilziel der Aufgabenstellung erreichen konnte
4. Der Nutzer hat Frust oder Verärgerung ausgedrückt, oder eine Änderung der Oberfläche vorgeschlagen
   [SaadawiEtAl2005]

Grundlage für diese automatische Anwendung sind Identifikationsmerkmale, die auch menschliche Beobachter verwenden (Vgl. [Nielsen1993]). Zwar kann das automatisierte Verfahren für diese Diplomarbeit nicht angewendet werden, da keine Thinking Aloud Protokolle vorliegen werden, allerdings konnten, basierend auf diesen Ansätzen, viele Usability-Probleme in den durchgeführten Sitzungen an folgenden Situationen leicht erkannt werden:

1. Teilnehmer fluchen laut
2. Teilnehmer schneiden eine Grimasse (aus den Webcam-Aufzeichnungen)
3. Teilnehmer stutzen / zögern bei der Bedienung (mehrere Sekunden ohne Interaktion mit dem System, „Denkpausen“)
4. Teilnehmer fragen andere Probanden (bzw. den Beobachter) um Hilfe, weil sie nicht weiter kommen
5. Probanden koordinieren sich untereinander (z.B. verbal) ohne dafür gedachte Funktionen des Systems zu nutzen

So konnte fehlende Workspace-Awareness gut am Verbalisieren der Probanden erkannt werden. Wenn ein Proband den anderen nach einer Information über dessen Aktivität oder über dessen Eclipse-Ansicht fragte, ohne hierbei Funktionen zu nutzen, die Saros dafür bereitstellt (z.B. weil er sie nicht entdeckt, oder nicht verstanden hatte), deutete dies auf ein Problem durch fehlende Übermittlung oder unverständliche Darstellung der Präsenz-Informationen hin.

Bei der Beobachtung der auftretenden Usability-Probleme eignet sich auch die Kategorisierung aus dem „Classification of Usability Problems (CUP) Scheme“ von Hvannberg und Law [CUP2003] als Grundlage für das Erkennen der Probleme, auch wenn es sich eigentlich um ein Schema für die später folgende Katalogisierung der Probleme handelt (siehe folgendes Kapitel).

In diesem Schema werden Probleme, die in Usability-Tests erkannt werden, in fünf Typen unterteilt.

- Missing (M)

  Der Nutzer findet etwas (z.B. eine Funktion oder Information) nicht, was da sein sollte.

- Incongruent mental model (I)

  Wenn die Oberfläche unklar ist, da sie nicht mit dem mentalen Modell des Nutzers übereinstimmt, oder mit den früheren Erfahrungen des Nutzers.

- Wrong (W)

  Es kann vorkommen, dass ein Nutzer merkt, dass etwas fehlgeschlagen ist (z.B. ein Programmierfehler)

- Better way (B)

  Wenn der Nutzer sich mit anderen Oberflächen/Systemen auskennt, kann der Nutzer mitteilen, dass etwas in der Benutzeroberfläche hätte besser gelöst werden können.

- Overlooked (O)

  Bei der Bearbeitung von Aufgaben kann es dazu kommen, das Nutzer die Funktion, die für die Bewältigung der Aufgabe gedacht ist, oder sich besonders dafür eignet, übersehen oder Nutzer nicht begreifen, dass sie mit dem Oberflächenelement interagieren sollen.

Diese Typisierung der Usability-Probleme zeigt auch Möglichkeiten auf, wie bei der Beobachtung vorgegangen werden kann. So sind verbale Äußerungen des Nutzers, z.B. wenn dieser erwähnt, dass in einem anderen System die gerade verwendete Funktion besser implementiert ist, ein guter Hinweis. Um zu bemerken, dass ein Nutzer eine Funktion übersieht (weil diese z.B. falsch platziert oder ungünstig benannt ist), ist eine Kenntnis des Systems notwendig, die bereits in der Durchführung der Heuristischen Analyse erlangt werden konnte. Das Auftreten von Fehlern zu bemerken, wird bei Kenntnis des Programms und der Implementierung leicht fallen. Sollte das System sich von den Erwartungen des Nutzers abweichend verhalten, ist die Wahrscheinlichkeit groß, dass dies kommentiert wird. Auch kann ein Stutzen bzw. Zögern des Nutzers darauf hindeuten, dass gerade Verständnisschwierigkeiten vorliegen (Vgl. [Nielsen1993]).

Mit diesen Beobachtungs-Leitlinien existierte somit eine Grundlage für die Beobachtung der Probanden bei der Nutzung von Saros. Um weitere Einblicke in das mentale Modell und die subjektive Zufriedenheit der Nutzer zu gewinnen,

bieten sich zusätzliche Verfahren, wie Interviews und Fragebögen an, deren Einsatzmöglichkeiten im folgenden Abschnitt beschrieben werden.

3.7 Interviews & Fragebögen

Interviews können dazu verwendet werden, nach einem Usability-Test die Nutzer nach ihren Erlebnissen zu befragen und daraus weitere Erkenntnisse über die Schwere der aufgetretenen Probleme zu gewinnen, und weitere Usability-Probleme abzuleiten. Viele Usability Probleme können schlicht durch Befragen von Nutzern festgestellt werden, insbesondere Probleme, die Auswirkungen auf die subjektive Zufriedenheit der Nutzer haben, die schwer objektiv zu messen wären [Nielsen1993,S. 209].

Es gibt unzählige Interviewtypen mit unterschiedlichem Standardisierungsgrad, von denen das episodische Interview am ehesten geeignet für den Einsatz im Rahmen dieser Diplomarbeit erscheint. Hierbei soll der Befragte frei erzählen, wobei zielgerichtete Fragen zu Erfahrungen, und daraus abgeleitetem Wissen des Befragten eingestreut werden können (Vgl. [Rudolph2011, S.7]).

Die Probanden werden nach den Usability-Tests gebeten, von Ihren Erlebnissen im Umgang mit Saros zu berichten, um hier einen Eindruck von deren subjektiven Empfindungen zu gewinnen und zu verstehen, wie diese die Oberfläche des Systems und dessen Verhalten interpretieren, und wo sie Verständnisprobleme hatten. Die Interviews sollten vor allem als Informationsquelle für ein besseres Verständnis des mentalen Modells der Nutzer dienen.

Bei der Durchführung des Interviews wurden hierbei folgende Leitfragen verwendet:

- Wie findest Du Saros (Subjektive Zufriedenheit)?
- Würdest Du Saros auch in Zukunft nutzen/weiterempfehlen?
- Hattest Du Probleme bei der Nutzung?
- Welche Probleme gab es?
    - An dieser Stelle wurden Probleme die von den Probanden erwähnt wurden aufgegriffen und genauer diskutiert, auch um mögliche Lösungsideen der Probanden zu ermitteln
- Welches Problem würdest Du als erstes beheben lassen, wenn Du dies bestimmen könntest?
- Welche Funktion fehlt Dir am meisten?

Von den vier Probanden konnten mit drei Probanden Interviews durchgeführt werden, mit einer Länge von etwa 30 Minuten. Die Ergebnisse dieser Interviews werden in Abschnitt 4.3, zusammen mit den sonstigen Analyseergebnissen, beschrieben.

Zusätzlich zu den Interviews wurden die Probanden gebeten, nach den Tests einen kurzen Fragebogen mit 9 Aussagen auf einer Likert-Skala[17] zu bewerten. Zur

[17] Die Likert-Skala dient der Messung persönlicher Einstellungen, die mittels so genannter Items abgefragt werden. (http://de.wikipedia.org/wiki/Likert-Skala)

Bearbeitung des Fragenbogens wurde ein kostenloses Konto bei www.surveymonkey.com verwendet.

Hierbei gab es folgende Antwortmöglichkeiten

- Ich stimme voll zu
- Ich stimme eher zu
- Weder noch
- Ich stimme eher nicht zu
- Ich stimme gar nicht zu

Die Aussagen des Fragebogens zielten vor allem darauf ab, die subjektive Einschätzung der Proband hinsichtlich bestimmter Systemeigenschaften zu ermitteln.

1. Ich empfinde den Einladungsprozess von Saros als kompliziert
2. Der Einladungsprozess von Saros ist schnell
3. Ich wusste zu jedem Zeitpunkt ob / das Saros gerade aktiv ist und etwas tut
4. Ich wusste zu jedem Zeitpunkt was genau (technisch) Saros gerade tut
5. Ich finde es gut wenn Saros mir in Fortschrittsbalken Details anzeigt, wie z.B. "Calculating Checksums for file: project/xyz.java"
6. Ich würde mir wünschen das Saros schneller arbeitet
7. Ich wusste zu jedem Zeitpunkt was die anderen Sitzungsteilnehmer gerade tun
8. Saros macht es mir leicht zu erkennen, welche Dateien, Ordner und Projekte Bestandteil der Sitzung sind
9. Ich würde Saros anderen Entwicklern empfehlen

Die Antworten auf die Fragebögen finden sich im Anhang 7.4, sowie in Abschnitt 5.4. Sollte keine Möglichkeit existieren, Anwender zu einem Interview zu überzeugen, können dennoch Einblicke in die Nutzung des Systems gewonnen werden, unter anderem durch Systemprotokolle und Nutzerbefragungen, z.B. durch webbasierte Fragebögen. Diese Methoden werden im folgenden Abschnitt vorgestellt.

3.8 Systemprotokolle, Nutzer-Feedback

Nielsen nennt „Logging actual Use“ (das Aufzeichnen tatsächlicher Nutzung) und „User Feedback“ als effiziente Methoden, um mehr über die tatsächliche Nutzung des Systems zu erfahren. Auch kann hierüber erkannt werden, wie die Nutzer ihre Aufgaben bewältigen. So können häufig genutzte Funktionen mit höherer Priorität verbessert werden, und Funktionen, die nicht genutzt werden, können sogar komplett entfernt werden (Vgl. [Sarodnick2006], [Nielsen1993, S.217]).

Saros stellt mit der Feedback-Komponente Mechanismen bereit, um nach Zustimmung des Nutzers, anonymisierte Nutzungsstatistiken zu einem Statistik-Server hochzuladen, der diese Daten aggregiert und mit einem regelmäßig laufenden Script eine Nutzungsstatistik generiert. Die Erkenntnisse aus dieser Nutzungsstatistik wurden bereits in Abschnitt 2.2 diskutiert.

Zusätzlich gibt es einen Saros-Fragebogen den Nutzer freiwillig ausfüllen können. Die Nutzer können nach jeder 5. Sitzung (außer sie haben dies deaktiviert) folgende Fragen frei beantworten:

- How did your latest Saros session work out? (Wie war deine letzte Saros Sitzung?)
- Which goals did you want to accomplish in your latest Saros session? (Was wolltest du bei der letzten Saros Sitzung erreichen?)
- What didn't go well (if anything) and what happened? (Was ist dabei schiefgegangen (wenn überhaupt)?)
- Do you have any other comments to share with us (e.g about the interface, specific features or possible improvements)? (Hast du sonst noch Kommentare zur Oberfläche, spezifischen Funktionen oder möglichen Verbesserungen)

Es gab insgesamt 78 ausgefüllte Fragebögen zwischen dem 03.07.2011 und dem 30.12.2011 (für aktuelle Saros-Versionen), wobei nur wenige der Rückmeldungen Hinweise auf Usability Probleme geben. Die Nutzer kommentieren vor allem, was sie mit Saros gemacht haben.

Diese Kommentare aus den Nutzerumfragen wurden unter anderem dazu herangezogen, die Frequenz der angesprochenen Probleme besser einschätzen zu können, und Hinweise auf weitere Probleme der Nutzer zu finden, die in Heuristischer Evaluation und Usability Test möglicherweise nicht entdeckt werden. Die anonymisierten Rohdaten der Nutzerumfragen finden sich unter [UmfragenWeb].

Die interessantesten Rückmeldungen in einem kurzen Überblick:

- Nutzer wünscht sich die Möglichkeit die Highlighting-Farbe der anderen Sitzungsteilnehmer ändern zu können, nicht nur die eigene
- Mehrere Nutzer wünschen sich unabhängig programmieren zu können und dann die Änderungen mit Saros zusammenzuführen (momentan überschreibt Saros die Änderung eines der Benutzer)
- Nutzer wünschen sich eine bessere Indikation, in welcher Datei sich ein Nutzer befindet, ohne „jump to position“ oder den Follow-Modus nutzen zu müssen
- Nutzer wünschen sich ein automatisches Ende von Sitzungen wenn nur noch ein Nutzer in der Sitzung verbleibt
- Verzögerungen im Chat werden kritisiert
- Mehrere Nutzern wünschen sich Möglichkeit farbliche Hervorhebung von Änderungen deaktivieren zu können
- Die Standard-Farben werden als zu schwach empfunden, das fehlende Speichern von durch Nutzer geänderter Farben wird kritisiert.
- Nutzer wünschen sich das Speichern von Selektionen, um nicht für jede Sitzung bestimmte Dateitypen (Binärdateien, Bilder etc.) manuell abwählen zu müssen

Wie zu erkennen ist, handelt es sich vor allem um Funktionswünsche, die auf mangelnde Prozessangemessenheit hindeuten können, insbesondere in Bezug

auf den Aspekt der Handlungsfreiheit (Vgl. [Sarodnick2006]). Die Nutzer wünschen sich häufig mehr Konfigurationsmöglichkeiten.

Nach dem nun festgestellt wurde, das eine Vielzahl an unterschiedlichen Verfahren existieren, und welche dieser Verfahren für die Analyse von Saros verwendet werden, bleibt zu erläutern, wie die identifizierten Probleme erfasst werden. Der Aufbau des Problemkatalogs wird im folgenden Abschnitt beschrieben.

3.9 Verfahren zur Katalogisierung der Probleme

Die Ergebnisse der durchgeführten Analysen sind Probleme, die in einem Problemkatalog erfasst werden, der Grundlage für die Überarbeitung von Saros darstellt. Neben einem prägnanten, kurzen Titel für Verweise auf das Problem, kommt eine inhaltliche Beschreibung, sowie die Angabe, welche Heuristiken verletzt wurden.

Zudem wird eine Gewichtung der Fatalität des Problems angegeben, deren Bestimmung die schwierigste Aufgabe darstellt, da diese sich aus drei komplexen Eigenschaften des Problems ableitet (Vgl. [Nielsen1993]):

- Häufigkeit des Auftretens (Frequenz)
    - Ist das Problem in einem Bereich der Software, der häufig genutzt wird, oder tritt es eher selten auf?
- Dauerhaftigkeit (Persistenz)
    - Kann der Anwender das Problem leicht umgehen, sobald er es erkannt hat? Gibt es einen alternativen Weg, oder ist die weitere Nutzung von Bestandteilen der Software verhindert?
- Ausmaß des (negativen) Einflusses auf die Aufgabenbewältigung
    - Wie sehr behindert das Problem den Anwender in seiner Arbeit? (z.B.: Wie groß ist die zeitliche Verzögerung durch dieses Problem?)

Die Gewichtung kann hierbei nach der Skala von Nielsen und Mack [Nielsen1994], mit Werten von 0 (kein Usability-Problem) bis 4 (Katastrophal), bestimmt werden, und dient später als Entscheidungsgrundlage für die Priorisierung bei der Behebung der Probleme.

0: Ich stimme nicht zu, dass es überhaupt ein Usability Problem darstellt

1: Kosmetisches Problem: Nur beheben wenn zusätzliche Zeit verfügbar ist

2: Kleineres Usability Problem: Beheben mit geringer Priorität

3: Großes Usability-Problem: Wichtig, daher mit hoher Priorität bearbeiten

4: Usability Katastrophe: Das Problem muss behoben werden, bevor das Produkt veröffentlicht werden kann

Die Gewichtung der Probleme ist besonders deshalb wichtig, da in einer heuristischen Evaluation viele geringfügige Probleme gefunden werden (der Anteil an allen gefundenen Problemen liegt bei etwa 72%, Vgl. [Nielsen1993]), deren Behebung viel Zeit in Anspruch nehmen kann, ohne dabei signifikante

Verbesserungen zu erzielen. Zwar argumentiert Nielsen in seinem Artikel zu „Severity Ratings in Heuristic Evaluation“ [Nielsen2000], dass die Gewichtung der Schwere eines Usability-Problems durch einen einzigen Evaluator wenig verlässlich ist, allerdings konnten die Ergebnisse früherer Arbeiten zur Usability für die Bewertung der Probleme herangezogen werden. Zudem wurden die Usability-Probleme in den wöchentlichen Treffen mit den Teammitgliedern diskutiert, und die Praxisrelevanz konnte anschließend in den Usability Tests untermauert werden.

Bei der Dokumentation der Probleme geht es nach Hvannenberg und Law (2003) nicht nur darum, die Probleme selbst zu beschreiben. Es macht auch Sinn, mögliche Ursachen für die Probleme zu ermitteln, und diese zu erfassen, um so eine Grundlage für Verbesserungen des Entwicklungsprozesses zu haben. Dies wird im „Classification of Usability Problems (CUP) Scheme“ von Hvannberg und Law [CUP2003] vorgeschlagen, die eine Struktur des Problemkatalogs mit insgesamt 13 Attributen vorstellen. In diesem Schema sind allerdings einige schwer verständliche und aufwändig zu ermittelnde Attribute beinhaltet, deren Bestimmung eine entsprechende Schulung bzw. umfassende Übung erfordert, die nicht gegeben war. Es beinhaltet insgesamt 13 Attribute, und unterscheidet dabei noch zwischen Einträgen im Problemkatalog für Probleme die in Usability-Tests ermittelt wurden, sowie Problemen, die in der heuristischen Evaluation ermittelt wurden.

Die für diese Arbeit verwendeten Attribute aus dem CUP Schema sind der „Trigger“ (Auslöser) - z.B. Verstöße gegen eine Heuristik oder beobachtete Probleme eines Nutzers in einem Usability Test sowie der „Impact“ (Auswirkung/Gewichtung), eine Gewichtung, die bestimmt, wie schwerwiegend das Problem ist, als Entscheidungsgrundlage für Entwickler, die die Reihenfolge der Problembearbeitung koordinieren müssen.

CUP verwendet bei der Gewichtung der Probleme nicht die 0-4 Skala von Nielsen und Mack. Es wird differenziert zwischen Ergebnissen aus der heuristischen Evaluation: „Minor, Moderate, Severe“, sowie Problemen aus Usability Tests: “effectiveness, efficiency, frustration expressed and help sought, either on-line or at a help-desk” [CUP2003, S. 3]

Die Einstufungen wurden hierbei wie folgt beschrieben:

- Severe: Probleme, die den Nutzer von der Bearbeitung der Aufgabe abhalten, oder katastrophalen Datenverlust oder Zeitverschwendung zur Folge haben
- Moderate: Probleme, die die Aufgabenerfüllung deutlich behindern, die aber umgangen werden können.
- Minor: Probleme, die verwirrend auf Nutzer wirken, aber nicht die Aufgabenerledigung behindern.

Die Auswirkungen (keine Gewichtung) in Usability-Test hingegen werden anders gewertet, in Hinsicht auf die Art des Problems: Mängel in der Effektivität, in der Effizienz, dem Ausdruck von Frustration durch den Nutzer oder dem Bitten um Hilfe durch den Nutzer.

Für diese Arbeit wurde auf eine explizite Differenzierung zwischen Problemen, die in einem Usability-Test auftreten, und Abweichungen von Heuristiken, verzichtet.

Die beiden Problemkataloge wurden in einen einzelnen Problemkatalog zusammengefasst. Dabei werden auch bei Problemen, die in einem Usability-Tests und nicht in der Heuristischen Evaluation gefunden wurden, Heuristiken referenziert, die verletzt wurden, bzw. die für die Problembehebung berücksichtigt werden sollten, da dies als wichtiger Hinweis für die Entwickler angesehen wurde.

Da bei der Erfassung der Usability-Probleme eine simple relationale MySQL-Datenbank eingesetzt wurde, um die Probleme später leichter zu katalogisieren, gewichten und auswerten zu können, ist es möglich, alle Komponenten zu recherchieren, die eine bestimmte Heuristik verletzten, und dabei auch die Probleme als Ergebnis zu erhalten, die neben der gewählten Heuristik noch weitere Heuristiken verletzen. So war es möglich, in der Datenbank eine problematische Komponente einmalig ganzheitlich zu beschreiben, sich aber bei der Bearbeitung der Probleme auch über die anderen Dimensionen orientieren zu können.

Aus eigener Erfahrung heraus ist es wahrscheinlich, dass ein Entwickler (insbesondere ich selbst) sich bei der Behebung von Problemen nicht alle Verletzungen einer bestimmten Heuristik vornehmen wird. Dafür wäre eine aufwändige Einarbeitung in mehrere unterschiedlichen Komponenten von Saros nötig, die das Kosten/Nutzen-Verhältnis negativ beeinflussen würde. Vielmehr werden Probleme aus dem jeweiligen Verantwortungsbereich (der eigens entwickelten Komponenten) bearbeitet. Daher wurden die Probleme zusätzlich nach Funktionsgruppen gegliedert. So ist es möglich sich beispielsweise mit allen Problemen aus dem Einladungsprozess zu beschäftigen, oder mit Problemen bei der Einrichtung von Saros.

Bei der Gewichtung der Probleme wurde sowohl die 0-4 Skala von Nielsen und Mack verwendet, als auch die simpleren und damit für Entwickler leichter zu interpretierenden Stufen: Severe, Moderate, Minor. Zudem wurden bei einigen Problemen die beiden Attribute Frequency (Auftrittshäufigkeit) und Persistence (Persistenz) mit erfasst. Dies ermöglicht es, besser nachzuvollziehen, wie eine meist stark subjektiv geprägte Gewichtung (Vgl. [Nielsen1993]) zustande kommt. Somit ist es für die Saros-Teammitglieder und in folgenden Abschlussarbeiten zum Thema Usability einfacher, die Gewichtung mit eigenen Einschätzungen abgleichen zu können. Auch sollte der Katalog so Saros-Entwicklern leichter zugänglich sein, da diese frei wählen können, welche Problemeinstufung ihnen am ehesten einen Hinweis darauf gibt, mit welcher Priorität das Problem bearbeitet werden soll.

Auf Basis der genannten Attribute aus dem CUP-Schema und eigenen Überlegungen ergibt sich folgendes Format für den Problemkatalog:

<ID>	**<Funktionsbereich>**	**<Titel>**	**<Gewichtungen>**	
<Beschreibung des Problems> **Heuristics:** <Verletzte Heuristiken> **Trigger:** <Auslöser, z.B: HE oder UT, Beschreibung des Problems der Probanden> **Typ:** <Typ des Problems im UT, z.B. Nutzer fragte nach Hilfe>, **Qualifier:** <Qualifier aus dem CUP-Schema, z.B. Missing (M)>				
	Änderung: <Was wurde verändert um das Problem zu lösen>			
	Lösungsansatz: <Möglicher Lösungsansatz zur Behebung des Problems>			

Die <ID> ist hierbei eine eindeutige Identifikationsnummer. Der Funktionsbereich ist ein Kategoriename, der angibt, in welchem Bereich des Systems das Problem auf trat (z.B. Einladungsprozess, Konfiguration). Dies soll die Orientierung für Entwickler erleichtern, die Expertise in bestimmten Funktionsbereichen von Saros haben und sich daher bei der Arbeit auf Probleme aus ihrem Bereich konzentrieren wollen. Dies erscheint mir besonders nützlich für die Abgrenzung der Probleme in experimentellen Funktionen, und den Problemen im stabileren Produktkern.

Der <Titel> beschreibt kompakt was das Problem ist. Neben der Gewichtung in Form von Worten und Zahlen wurde noch eine Balkendarstellung hinzugefügt, die die Gewichtung von Problemen zwischen 1 und 4 mit schwarzen Balken visualisiert.

Der <Typ> des Problems (siehe vorherigen Abschnitt „Identifikation der Probleme als Beobachter eines Usability-Tests“) kann folgende Werte annehmen: Missing (M), Incongruent mental model (I), Wrong (W), Better way (B), Overlooked (O).

Falls das System überarbeitet wurde um das Problem zu lösen ist bei <Änderung> dokumentiert, was verändert wurde.

Falls das Problem nicht vollständig gelöst werden konnte, oder nicht bearbeitet wurde, kann ein <Lösungsvorschlag> angegeben sein. Hier können sowohl konkrete Implementierungsansätze erläutert sein, als auch abstrakte Konzepte.

Damit Entwickler sich im Katalog inhaltlich schnell orientieren können, wurden die gefundenen Probleme in folgende Nutzungs-Bereiche eingeteilt:

- Einrichtung
- Einladungsprozess (Starten einer Sitzung)
- Laufender Einladungsprozess
- Laufende Saros-Sitzung
- Ende/Beenden einer Saros-Sitzung

Zudem gibt es noch Gruppen für „allgemeine Probleme“ in Wizards/Dialogen/Meldungen/Views sowie „Experimentelle Funktionen“, darunter

Screensharing und Voice-over-IP. Die Probleme lassen sich hierbei nach allen Attributen sortieren, darunter auch nach der Gewichtung des Problems. Somit kann bei der Bearbeitung eine Sortierung gewählt werden, die dabei hilft, schwerwiegende Usability-Probleme zuerst zu behandeln.

Im Anhang 7.7 findet sich aus Platzgründen nur eine einfache Darstellung des Katalogs. Die Probleme sind dort gruppiert nach den Funktions-/Nutzungsbereichen und absteigend sortiert nach der Gewichtung der Probleme.

Nach dem in diesem Kapitel vorgestellt wurde, welche Methoden zur Verfügung stehen, und ob diese geeignet sind für die Analyse und Überarbeitung von Saros, wird im nächsten Kapitel beschrieben, welcher Plan für die Analyse aufgestellt wurde, die anschließende Durchführung der Tests wird dokumentiert, und es wird diskutiert welche Auswirkungen die eher unkontrollierte Testumgebung auf Reliabilität und Validität der Ergebnisse hat.

4 Durchführung der Analyse

Nachdem die Verfahren vorgestellt und hinsichtlich ihrer Eignung für den Einsatz bei der Analyse von Saros betrachtet wurden, wird in diesem Kapitel die durchgeführte Analyse beschrieben. Nach einer Übersicht über die Testumgebung und die verwendeten Software- und Hardware-Ausrüstung, werden die Eigenheiten der Probanden und deren Einfluss auf die Ergebnisse dokumentiert. Zudem werden die bearbeiteten Aufgabenstellungen und die aufgezeichneten Testszenarien beschrieben. Da es nicht möglich war, alle Probanden gleichzeitig zu beobachten, war es notwendig Aufzeichnungen der Tests anzufertigen. Die dazu verwendete Technik wird ebenfalls in diesem Kapitel beschrieben.

Der Empfehlung von Sarodnick und Brau (2006), und Nielsen (1993) folgend, wurde die Analyse der Usability in einem mehrstufigen Prozess durchgeführt, der auch schrittweise Überarbeitungen von Saros beinhaltet. Insbesondere wurde das Schlüsselkonzept, den frühen und schnellen Iterationen, mit häufigen Rückmeldungen über die Usability, angewandt. (Siehe auch „Discount Usability Engineering“ von J. Nielsen und „Iterative Design“ [Tidwell2011]):

1. In einer heuristischen Evaluation werden die Oberfläche und das Verhalten von Saros mit den 10 gängigen Heuristiken von [Nielsen1994] verglichen. Die gefundenen Abweichungen zu den Heuristiken, und die daraus möglicherweise folgenden Usability-Probleme, werden katalogisiert und vorläufig priorisiert.
2. Probleme, bei denen eine Priorisierung schwer fällt, werden im Saros-Team diskutiert, um die Priorisierung zu präzisieren. Bei der Analyse gefundene Software-Fehler werden im Sourceforge-Bugtracker erfasst und Lösungen diskutiert.
3. Software-Fehler, die die erfolgreiche Durchführung der Usability-Tests gefährden könnten, werden vor der Durchführung beseitigt, oder es wird ermittelt, wie diese bei der Durchführung der Tests umgangen werden können
4. In Usability-Tests wird die Usability von Saros im Praxiseinsatz in realistischem Umfeld analysiert. Die Protokolle der Sitzungen, sowie die Videoaufzeichnungen werden ausgewertet und der Problemkatalog wird mit den gefundenen Problemen aktualisiert.
5. Anhand der Frequenz und Persistenz der Probleme, die in den Usability-Tests entdeckt wurden, wird die Gewichtung der Probleme präzisiert. Vermutete Usability-Probleme, die sich aus der Heuristischen Evaluation ergaben, werden so bestätigt.
6. In der vorgegebenen Zeit voraussichtlich erfolgreich zu behebende Probleme, werden aus dem Problemkatalog ausgewählt und Lösungskonzepte werden erstellt, die mit dem Saros-Team diskutiert werden.
7. Prototypen werden entwickelt, um die Usability der Lösungskonzepte frühzeitig zu prüfen und so zu verhindern, dass z.B. komplexe Benutzeroberflächen implementiert werden, die sich dann im Praxiseinsatz als unbrauchbar herausstellen. Hierbei werden primär interaktive

„Wireframes“[18] eingesetzt. Schließlich werden die erfolgreich evaluierten Konzepte implementiert.

8. Neue prototypische Versionen von Saros werden in weiteren Usability-Tests evaluiert, um zu prüfen, ob die beobachteten Usability-Probleme tatsächlich zufriedenstellend behoben wurden, und ob ggf. neue Usability-Probleme entstanden sind, oder Probleme, die bisher verdeckt waren, zum Vorschein kommen.
9. Schritte 6 bis 8 werden so oft wie möglich wiederholt.

Nielsen empfiehlt in seinem Paper “Iterative User Interface Design” [Nielsen1993p] wenigstens drei Iterationen. Da die Verbesserung der Usability meist auf einen Teilaspekt der Software fokussiert erfolgt, kann es dazu kommen, dass durch die Änderungen die Usability der Software in anderen Bereichen verschlechtert werden kann. Da selbst der beste Usability Experte nicht in der Lage sei eine perfekte Benutzeroberfläche in nur einem Versuch zu erstellen, sollte ein Usability Engineering Lebenszyklus stets Iterationen aus Implementierung und Evaluation durch Nutzer-Tests beinhalten, um die Qualität der Ergebnisse zu verbessern.

Die Implementierung von Lösungskonzepten, sowie die anschließende Evaluierung von Prototypen, wurden daher in mehreren Iterationen durchgeführt.

Im folgenden Abschnitt werden die Durchführung der Usability-Tests und die resultierenden Ergebnisse beschrieben, und mögliche Probleme in Bezug auf die Aussagekraft der Ergebnisse werden aufgedeckt.

4.1.1 Testumgebung

Eine Usability-Labor-Umgebung war bei den Usability-Tests vor Ort im Unternehmen nicht gegeben. Die Tests erfolgten direkt an den Rechnern der Entwickler, die sonst für die tägliche Arbeit verwendet wurden. Auch wurden die spezifischen Eclipse-Versionen der Entwickler verwendet (die z.B. Komponenten des Java-Spring-Frameworks beinhalten, was mögliche Inkompatibilitäten mit Saros zur Folge haben könnte).

Die Testumgebung konnte nur dahingehend kontrolliert werden, dass vor dem Start der Usability-Tests die Eclipse-Versionen vorbereitet wurden und geprüft wurde, ob diese trotz Unternehmens-Firewall[19] und Proxy[20] funktionsfähig sind.

Diese Kombination aus kontrollierten Usability-Tests und Feldstudie kann sowohl negative, als auch positive Auswirkungen haben.

Die Ausstattung mit Computer-Hardware kann nicht beeinflusst werden. Mögliche negative Auswirkungen wären mangelnde Realitätsnähe durch zu hohe Leistung der verwendeten Rechner und Netzwerkverbindungen im Vergleich zur

---

[18] Sehr früher konzeptueller Prototyp eines Software-Frontends (Ähnlich einer Strichzeichnung, meist schwarz/weiß)

[19] Software die Netzwerkzugriffe beschränkt und kontrolliert

[20] Kommunikationsschnittstelle im Netzwerk, arbeitet als Vermittler für Verbindungsversuche z.B. zu externen Systemen. Kann die Kommunikation beeinflussen und z.B. bestimmte Arten von Anfragen verhindern

Ausstattung der tatsächlichen Saros-Nutzer. Die Tests erfolgten mit etwa 2-3 Jahre alten Computern, mit Mehrkern-Prozessoren und 4-8 GB Arbeitsspeicher, sowie herkömmlichen Festplatten mit beweglichen Teilen (keine Solid State Disks). Da keine Informationen über die technische Ausstattung der tatsächlichen Saros-Anwender vorliegen, kann hier nicht weiter verglichen werden, ob dies Einfluss auf die Ergebnisse hat.

Das Firmennetz ist ein 1 GBit/s Netzwerk. Die Geschwindigkeit der Internetverbindung konnte nicht genau bestimmt werden, kann aber nach eigenen Messungen bei bis zu 100MBit/s (je nach Netzwerkauslastung) vermutet werden. Auch die externen Unternehmensstandorte sind damit mit einer hohen Netzwerkgeschwindigkeit angebunden, die sonst nur in lokalen Netzen verfügbar ist.

Usability- und sonstige Probleme durch eine langsame Datenübertragung bei den Beobachtungen können daher nicht ermittelt werden. Eine Installation von Software zur Regulierung der Verbindungsgeschwindigkeit auf etwa 80 KB/s Übertragungs-Geschwindigkeit (entspricht etwa DSL 16.000) wurde getestet, allerdings scheiterte dies an Inkompatibilitäten zu anderer installierter Software (vermutlich die VMWare-Treiber) und die Rechner der Entwickler stürzten mit einem BlueScreen[21] ab. Da das Betriebssystem danach nur noch im abgesicherten Modus gestartet werden konnte, und die Aufregung der Entwickler entsprechend groß war, musste somit diese Software wieder deinstalliert werden.

Da es nicht möglich war, mit den Probanden zu testen, welche negativen Auswirkungen die Datenübertragungsgeschwindigkeit hat, wurden eigene Tests durchgeführt, mit einem Desktop-PC und einem Laptop, die mit entsprechender Geschwindigkeitsdrosselung ausgestattet wurden. Dies waren zwar keine repräsentativen Usability-Tests, aber sie erlaubten zumindest grundlegende Rückschlüsse auf mögliche Probleme bei der Datenübertragung. Zudem lagen Erkenntnisse aus den Auswertungen der Videoaufzeichnungen aus der Arbeit von Arsenij Solovjev [Solovjev2011] vor, bei denen Nutzer in Tests mit einer IBB-Verbindung arbeiteten, die gemäß der Saros-Nutzungsstatistik eine durchschnittliche Datenübertragungsgeschwindigkeit von nur ca. 13KB/s aufweist.

Aus der Analyse der Nutzungsstatistik ergab sich, dass die durchschnittliche übertragene Datenmenge bei 3.25MB liegt, womit selbst bei sehr langsamer Verbindung die Übertragung kein Problem darstellt, da dies höchstens 5 Minuten dauern würde.

4.1.2 Setup und technische Ausrüstung

Aufgezeichnet wurden in den einzelnen Sitzungen sowohl der Bildschirminhalt jedes Probanden, als auch der Ton über bereitgestellte Headsets. Sofern der Entwickler eine der bereitgestellten Webcams angeschlossen hatte, wurden auch Aufnahmen des Entwicklers selbst vorgenommen.

---

[21] Beschreibung einer Kategorie von kritischen Fehlermeldungen, die z.B. von Windows-Betriebssystemen auf blauem Hintergrund dargestellt werden

Die Probanden telefonierten während der Tests über die VoIP-Software Skype miteinander, da die Voice-over-IP Integration in Saros nicht zuverlässig funktionierte, und die Entwickler von vorneherein an die Nutzung von Skype als Kommunikationsmittel gewöhnt waren.

Zur kombinierten Aufzeichnung von Bildschirminhalt, Webcam und Ton wurde die Software von www.screencast-o-matic.com verwendet. Diese Software kombiniert die einzelnen Video-/Ton-Ströme miteinander in eine einzelne Videodatei (pro Proband).

Als besonders praktisch stellte bei dieser Software heraus, dass Mausklicks in den Videos durch einen sich langsam ausblendenden Kreis um den Mauszeiger hervorgehoben werden, sodass nachvollziehbar ist, wann und wo der Proband auf dem Bildschirm geklickt hat. Dies erleichterte die spätere Auswertung der Videoaufzeichnungen deutlich.

4.1.3 Aufgabenstellungen

Die Tests wurden gemäß den Empfehlungen von Nielsen (1993) durchgeführt. Die Probanden wurden vorab über die konkrete Aufgabenstellung informiert, sofern sie nicht selbst eine Aufgabe vorliegen hatten (z.B. ein zu erledigender Code-Review). Die Probanden wurden während der Bearbeitung der Aufgaben nur in absoluten Ausnahmefällen unterstützt, z.B. als einer der Probanden aufgrund von Systemeinstellungen an der Einrichtung scheiterte.

Sie sollten das System eigenständig erkunden, weshalb es keine Vorstellung aller Funktionen des Systems gab, auch um den Nutzern keinen Navigationspfad vorzuführen, z.B. durch Nutzung bestimmter Menüs, der dann mit großer Wahrscheinlichkeit nachgeahmt werden würde. Den Probanden wurde erklärt, wozu das System gedacht ist (gemeinsame Programmierung, Code-Reviews), aber nicht, welche Funktionen es genau es hat. Dies sollten die Probanden eigenständig herausfinden, auch um die Usability-Eigenschaft „Erlernbarkeit" beurteilen zu können. Die Probanden wurden informiert, dass sie sich auf der Webseite Informationen über den Funktionsumfang von Saros verschaffen können, allerdings hat keiner der Probanden diese Möglichkeit genutzt.

Die vorhandenen Eclipse-Versionen der Probanden wurden mit der jeweils zu testenden Saros-Version präpariert, sodass der schwerfällige Installationsprozess nicht von den Probanden vorgenommen werden musste. Der Installationsprozess ist definitiv problematisch, allerdings war ein Test der Installation nicht das Ziel der Analyse. Auch hätte dies signifikant viel Zeit in Anspruch genommen. Da die Probanden nur mit einem begrenzten Zeitbudget zur Verfügung standen, wurde Wert darauf gelegt, derart langwierige Prozesse zu vermeiden.

Für die Tests mit vorgegebenen Aufgabenstellungen wurden Aufgaben aus dem „CodingDojo"[22] gewählt, einer Webseite, die Problemstellungen beinhaltet, die Programmierer bei Veranstaltungen gemeinschaftlich bearbeiten, um Ihre Fertigkeiten als Entwickler zu prüfen und zu verbessern. Ein Fokus bei der

[22] http://codingdojo.org/cgi-bin/wiki.pl?KataCatalogue

Bearbeitung dieser Aufgaben liegt hier darin, die individuelle Eignung zur Paarprogrammierung zu ermitteln.

Die Aufgaben sind dabei stets simpel gehalten, und benötigen für die Bearbeitung meist weniger als 30 Minuten, was zu den Usability-Tests passt. Es wurden folgende Aufgabenstellungen bearbeitet:

1. „KataFizzBuzz“: Es werden Zahlen von 1 bis 100 auf der Ausgabekonsole ausgegeben, wobei statt Zahlen die durch 3 teilbar sind der Text „Fizz“, bei Zahlen die durch 5 teilbar sind der Text „Buzz“ und bei Zahlen die durch 3 und 5 teilbar sind der Text „FizzBuzz“ ausgegeben wird.
2. „KataRomanNumerals“: Dezimalzahlen sollen in römische Ziffern umgerechnet werden. So ergibt 1990 z.B. „MCMXC“. Die Zahlen von 1 bis 100 sollen auf der Konsole ausgegeben werden, „1999“ soll als Akzeptanz-Test verwendet werden
3. „KataWordWrap“: Eine Zeichenkette aus Worten ohne jegliche Zeilenumbrüche soll so formatiert werden, dass keine Zeile länger als 80 Zeichen lang ist, damit diese in einem Emailprogramm auf kleinem Monitor gut dargestellt werden kann. Dazu werden Zeilenumbrüche eingefügt, ohne dabei Worte zu trennen. Es soll so umgebrochen werden, dass lange Wörter, die nicht mehr auf die aktuelle Zeile passen in der nächsten Zeile landen und nicht in der Mitte eines Wortes ein Zeilenumbruch eingefügt wird.

Diese Aufgaben wurden vor den Tests auf Ihre Bearbeitungszeit geprüft, in dem sie eigenständig durchgeführt wurden, und dabei die Zeit gestoppt wurde. Für keine Aufgabe wurde dabei länger als 30 Minuten benötigt. Somit war die Wahrscheinlichkeit hoch, das auch bei unterschiedlicher Qualifikation der Probanden und unterschiedlichen Arbeitsgeschwindigkeiten, eine zumindest teilweise erfolgreiche Erledigung der Aufgaben möglich ist (was ein Erfolgserlebnis für die Probanden darstellen würde). Bei den Tests wurde es allerdings nicht als entscheidend angesehen, ob die eigentliche Programmier-Aufgabe komplett gelöst wurde.

Bei den durchgeführten Tests wurden keine konkreten Aufgaben vorgegeben, die sich auf Funktionen von Saros beziehen. Vielmehr lag der Fokus bei der Aufgabenstellung darin, die Probanden dazu zu bringen, selbst zu entscheiden, wie sie Saros zur Lösung des Problems einsetzen, und welche Funktionen sie dabei entdecken und nutzen. Dadurch erhoffte ich mir Rückschlüsse auf die Erlernbarkeit von Saros, und Informationen über die Vorgehensweise der Nutzer bei der Erkundung des Systems, und es sollte dabei herausgefunden werden, welche Funktionen ihnen verborgen bleiben. Zudem sollte dadurch verhindert werden, dass (möglicherweise schwerwiegende) Probleme in Funktionen ermittelt werden, die keine Praxisrelevanz haben (z.B. im VoIP-Modul).

So konnte z.B. ermittelt werden, dass die Funktion „Partial Sharing“ den Nutzern unerschlossen blieb, vermutlich da die Oberfläche von Saros nicht geeignet auf diese Funktion hinweist.

4.1.4 Test-Szenarien

Die Usability-Tests sollten auf Basis der Erkenntnisse aus der Statistikanalyse und den Nutzerumfragen insgesamt folgende Nutzung von Saros abdecken:

- Sitzungen zu zweit, mit Nutzung des Follow-Modus
- Sitzungen zu dritt, mit Nutzung des Follow-Modus
- Gemeinsames Editieren in der selben Datei
- Editieren in unterschiedlichen Dateien

Die Nutzungsstatistik ergibt zwar, dass es fast nur Sitzungen mit 2 Teilnehmern gibt, allerdings kann dies auch darin begründet sein, das Saros für die Nutzung mit 3 Teilnehmern ungeeignet ist. So bestätigte sich schon in der Heuristischen Evaluation, dass es Probleme durch fehlende Awarenessinformationen gibt. Eine Bestätigung für die Bedeutsamkeit des Fehlens von Informationen wie z.B. „Wer folgt wem in der Sitzung“ und „wer hat welche Datei offen“ sollte hier eingeholt werden.

Die tatsächlichen Aufzeichnungen der Saros-Nutzung im Unternehmen beinhalteten folgende Szenarien, die teils von den Entwicklern inspiriert waren, und teils vorgeben wurden, um Kontrolle über den Testverlauf zu gewinnen:

- (Pilot-Testlauf) Interaktiver Code-Review mit 3 Teilnehmern (1 Driver, 2 Observer)
- (Pilot-Testlauf) Code-Review mit 2 Teilnehmern
- (Pilot-Testlauf) Paarprogrammierung mit 2 Teilnehmern
- Code-Einführung in JUnit-Tests mit 2 Teilnehmern
- Code-Review & Paarprogrammierung gemischt mit 2 Teilnehmern
- Saros-Tutorial: ein erfahrener Nutzer erklärt einem anderen unerfahrenen Nutzer wie Saros funktioniert und bedient werden sollte (2 Teilnehmer)
- Paarprogrammierung mit 2 Teilnehmern
- Code-Review, dann Wechsel zu Paarprogrammierung in einer Sitzung mit 2 Teilnehmern

Besonders hervorzuheben ist an dieser Stelle noch die Motivation für das Szenario, bei dem ein erfahrener Proband, der Saros bereits mehrfach eingesetzt hat, einem anderen Probanden erklären sollte, wie Saros funktioniert, mit einer kurzen Beispiel-Code-Einführung zur Demonstration der Saros-Funktionen. Hierdurch erwartete ich, dass sich Möglichkeiten offenbaren, Einblicke in das mentale Modell der Nutzer zu erhalten und dies für weitere Analysen und Optimierung verwenden zu können.

Um vor den Tests zu prüfen, ob die Aufzeichnungs-Technik einwandfrei funktioniert und um sicherzugehen, dass auch getestet wird, wie die Usability beim Einsatz mit 3 oder mehr Teilnehmer ist, wurden mehrere Pilot-Tests durchgeführt, darunter einen Code-Review mit 2 Teilnehmern, bei denen ich wechselnd als Driver und als Observer tätig war. Auch dieser Erstkontakt der Probanden mit Saros wurde aufgezeichnet und protokolliert.

4.1.5 Tests von Prototypen

Bei den Tests der Prototypen von Saros wussten die Probanden zwar, dass eine neue Version von Saros getestet wird, allerdings wurden sie vorher nicht darüber informiert, welche Änderungen genau erfolgt sind.

Dies unterscheidet sich von der realen Nutzung von Saros dahingehend, dass bei einem neuen Saros-Release in der Ankündigung stets ein Changelog mit ausgeliefert wird, das von Nutzern gelesen werden könnte. Da aus eigenen Erfahrungen heraus, nicht jeder Nutzer das Changelog tatsächlich aufmerksam liest, bevor die neue Version getestet wird, wird kein negativer Einfluss auf die Probanden erwartet. Auch hätte sonst die Möglichkeit bestanden, dass die Nutzer durch euphorische Beschreibung der durch mich vorgenommenen Verbesserungen, diese eher wahrnehmen und positiver bewerten würden, als dies sonst der Fall wäre (Vgl. „Priming“ [WikiPriming]).

Die ersten getesteten Prototypen beinhalteten nur Verbesserungen bezüglich der Darstellung von Awareness-Informationen, darunter die Informationen wer wem folgt, und wer der Host der Sitzung ist. Insbesondere sollte hier zunächst evaluiert werden ob Nutzer die neuen Informationen schnell entdecken, die Informationen verwenden und ob dies die verbale Koordination in den Sitzungen verringern kann, bevor dieser Ansatz weiter verfolgt wird.

Sarodnick und Brau (2006) erwähnen, dass die Behebung von Problemen die von Nutzern explizit bemängelt werden, am relevantesten für die Nutzerzufriedenheit sind. Da die gravierendsten Probleme in Bezug auf die Effizienz der Arbeit mit Saros, im Bereich mangelnder Awareness lagen, wurden diese Probleme zuerst bearbeitet.

Auch stellte sich schnell heraus, dass die geplanten Änderungen an den Verbesserungen zum System-Feedback (Heuristik N1) schwieriger als gedacht sind. Somit konnten diese Veränderungen erst in den darauffolgenden Prototypen geprüft werden. Diese beinhalteten zudem Veränderungen in den Beschriftungen und der Anordnung der Elemente, sowie Performance-Optimierungen im Einladungsprozess.

Nach dem in diesem Abschnitt beschrieben wurde, welche Tests durchgeführt wurden, und mit welchen Ressourcen und welcher Technik, werden im folgenden Abschnitt die Ergebnisse der Evaluation präsentiert. Anschließend erfolgt eine Betrachtung der Reliabilität und Validität.

4.2 Zusammenfassung der Analyse-Ergebnisse

Die Ergebnisse aus der heuristischen Analyse, sowie den Usability-Tests in diesem Abschnitt, beinhalten Erkenntnisse aus der Beobachtung der Probanden, den Auswertungen von Videoaufzeichnungen, sowie Ergebnissen aus der Analyse der Videoaufzeichnungen von Arsenij Solovjev aus [Solovjev2011]. Ebenso flossen die Analyseergebnisse aus der Masterarbeit von Björn Kahlert zur Out-of-Box-Experience [Kahlert2011], und die Rückmeldungen von Nutzern, über den in Saros-integrierten Umfrage-Mechanismus, in die Beschreibung der Schwachstellen in Saros mit ein.

Sarodnick und Brau (2006) beobachteten, dass es kaum möglich ist alle Daten, die bei einer Evaluation anfallen, wissenschaftlich korrekt auszuwerten. Anstatt also den Problemkatalog im Detailgrad weiter auszubauen, wurde der Fokus auf die Überarbeitung der Usability gelegt. Dennoch erfolgt zunächst eine zusammenfassende Betrachtung der Probleme. Die Beschreibung aller 128 Probleme findet sich im Problemkatalog im Anhang 7.7.

Die Beobachtungen der Probanden waren aufschlussreich und die Ergebnisse aus [Kahlert2011] und [Solovjev2011] konnten bestätigt, aber auch um weitere wichtige Usability-Probleme ergänzt werden.

- Die Nutzer sind begeistert von der Idee hinter Saros, bemängeln aber die Schwächen in der Umsetzung. Insbesondere die als kompliziert empfundene Einrichtung, sowie der langsame Einladungsprozess, senken die subjektive Zufriedenheit der Nutzer. Nutzer wissen meistens nicht, was Saros tut, und insbesondere nicht, wie lange es noch dauern könnte.
- Durch mangelnde Awarenessinformationen wird der Arbeitsfluss der Nutzer gestört. Nutzer hinterfragen die Tauglichkeit von Saros für ihre Einsatzzwecke aufgrund dieser Mängel.
- Eine bessere Systemintegration in bestehende IT-Landschaften wird gewünscht (z.B: Import von Skype-Kontakten, Nutzung von Hostnamen in geschlossenen Firmennetzwerken).
- Saros weist Schwächen bei der Erlernbarkeit auf. Die Nutzer konnten die Existenz der Funktion Partial Sharing nicht erkennen, nutzten diese aber begeistert, nach dem sie darauf hingewiesen wurden.
- Der Entwicklungsstand von Saros wurde von den Probanden einstimmig als „alpha"-Version angesehen.

Um einen ersten Eindruck von der Verteilung der identifizierten Usability-Probleme zu gewinnen, wurden diese nach den verletzten Heuristiken gruppiert. Bei Usability-Problemen, die nicht in der heuristischen Evaluation identifiziert wurden, wurden nachträglich Heuristiken zugewiesen, die verletzt wurden bzw. für die Behebung des Problems herangezogen werden können.

Wenn ein Element in der Oberfläche von Saros gleich mehrere Heuristiken verletzt, wird es jeweils einmal gezählt pro Heuristik, die verletzt wird. Somit ergeben sich aus 128 identifizierten Problemen insgesamt 203 Heuristik-Verletzungen.

Die Höhe der Farbbalken in den Diagrammen gibt dabei die Anzahl der Probleme in diesem Bereich wieder. Die Farbunterteilung ist wie folgt:

- 1 - Blau = Kosmetisches Problem
- 2 - Gelb: Kleineres Usability Problem
- 3 - Orange = Größeres Usability Problem
- 4 - Rot = Schweres Usability-Problem (Katastrophe)

Hierbei ist zu beachten, dass beispielsweise eine Usability-Katastrophe wesentlich bedeutsamer ist als vier kosmetische Probleme. Die Skala von Nielsen und Mack stellt kein absolutes Verhältnis zwischen den Problemen dar.

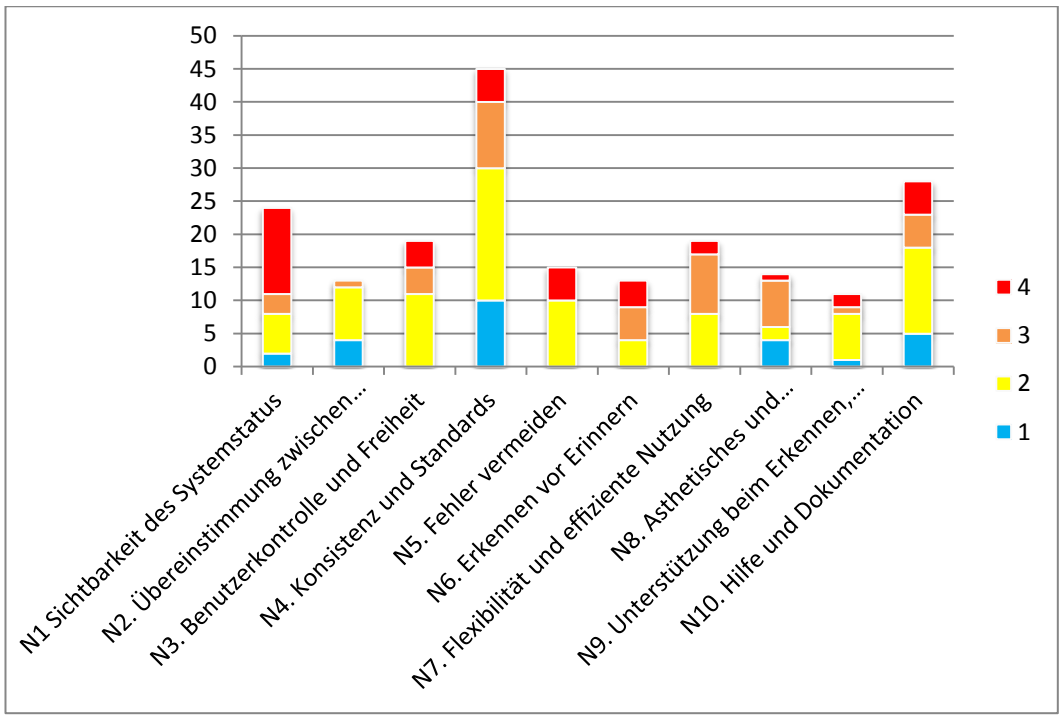

Abbildung 6: Usability Probleme nach Heuristik-Verletzungen

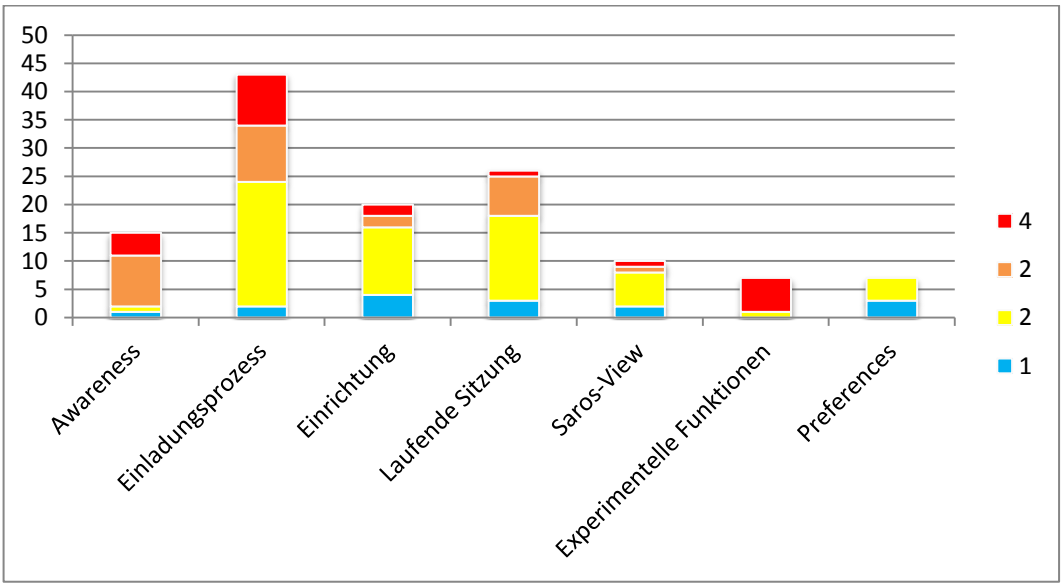

Abbildung 7: Usability Probleme nach Funktionsbereichen

Wie zu erkennen ist, ist vor allem der Einladungsprozess problematisch. Auch die experimentellen Funktionen beinhalten viele kritische Usability-Probleme. Bei der Entwicklung von Saros wurde nicht auf die Einhaltung von Standards geachtet, und die Oberfläche und das Interaktionskonzept sind an vielen Stellen nicht konsistent. Saros weist Mängel in der Erwartungskonformität, durch geringe Konsistenz auf, was negativ auf die Lernförderlichkeit wirkt (Vgl. [Dix2003]). Nutzer entdecken viele der Funktionen auch in weiteren Tests nicht. Informationen über den Systemzustand gibt es so gut wie keine, und wenn, sind diese ungeeignet durch technische Formulierungen.

Im Folgenden werden nun die identifizierten Schwachstellen präsentiert, wobei diese dabei in Problembereiche eingeteilt sind.

- Einrichtung
- Aufbau einer Sitzung (Einladungsprozess)
- Laufende Sitzung
- Awareness, Saros-View (allgemeine Darstellung, Interaktionen)
- Experimentelle Funktionen (Screensharing/VoIP)

Neben der Erwähnung von besonders bedeutsamen Usability Problemen, die z.B. in den Usability-Tests ermittelt wurden, wird für jede Art von Heuristik-Verletzung oder beobachteten Probleme jeweils exemplarisch ein Problem vorgestellt, um einen Eindruck vom Gesamtzustand des Systems, sowie der subjektiven Zufriedenheit der Nutzer mit dem System zu geben. Der vollständige Problemkatalog mit einer Beschreibung aller identifizierten Usability-Probleme findet sich im Anhang 7.7.

4.2.1 Probleme bei der Einrichtung

Bei der Einrichtung von Saros hatten die Probanden unter anderem Probleme damit, das sich ihnen nicht erschloss, wieso sie eine XMPP-ID benötigen, und äußerten Wünsche, dass sie sich gerne mit der Skype-ID verbinden würden, da sie bereits Skype einsetzten. Auch wurde hinterfragt, wieso Saros im Firmennetz nicht mit IP-Adressen oder Hostnamen arbeiten kann.

Generell wurde eine stärkere Integration in bestehende Systeme gewünscht. So erschien es den Nutzern umständlich, zunächst alle Arbeitskollegen als Buddy hinzuzufügen, obwohl diese z.B. in Skype oder Outlook bereits als Kontakte hinterlegt waren. Integrationen in technische Umgebungen wie z.B. Active Directory, Microsoft Exchange Server und co. fehlen in Saros vollständig.

Der „Getting Started“ Wizard, der neuen Nutzern den Einstieg in das System erleichtern soll, wurde von den Probanden einstimmig als „unpraktisch“ empfunden. Während einige Nutzer zwar versuchten, die Inhalte aus den dort enthaltenen Screenshots zu verstehen, „überblätterten“ andere Nutzer den Wizard komplett. Die Nutzer, die die Grafiken genauer betrachtet hatten, waren bei der Einrichtung von Saros danach nicht schneller als die Nutzer, die die Einführung übersprungen haben. Auch bemängelten Nutzer in Befragungen, direkt im Anschluss an die Usability-Tests, dass die Einführung keine Informationen über den Funktionsumfang von Saros bereitstellt. Aus weiteren Erfahrungsberichten der Diplomarbeits-Betreuerin kam hinzu, dass offenbar Probanden versucht hatten, mit den Screenshots zu interagieren.

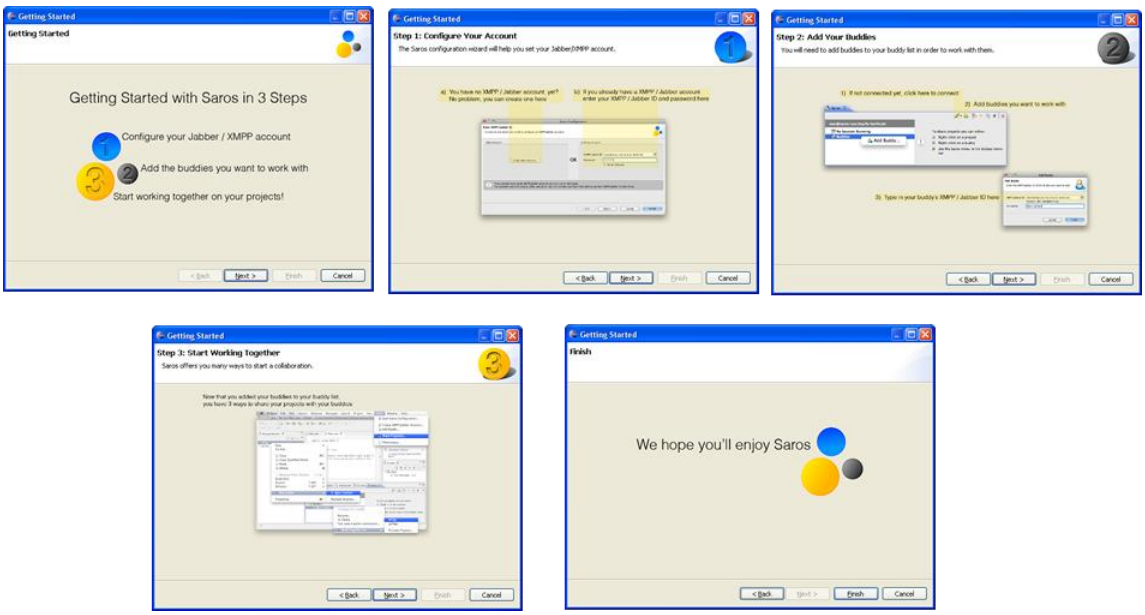

Abbildung 8: Der Getting Started Wizard

Je nach Bildschirmauflösung gab es zudem Probleme mit der Darstellung der Dialoge/Wizards in Saros. So wird der Getting Started Wizard in prozentualer Größe zur Bildschirmauflösung eingeblendet. Hierdurch sind die Texte in den Screenshots nicht mehr lesbar, da die Grafiken verkleinert dargestellt werden.

Da fest einprogrammierte Größen (Höhe und Breite) für alle Wizards vergeben wurden, ist Eclipse nicht in der Lage das Wizard-Fenster automatisch zu vergrößern, wenn der Nutzer eine Windows-Einstellung zur Erleichterung der Lesbarkeit auf dem Bildschirm verwendet, die die Titelleiste des Fensters und bestimmte Interaktionselemente vergrößert darstellt (120%, bzw. 150% der Originalgröße).

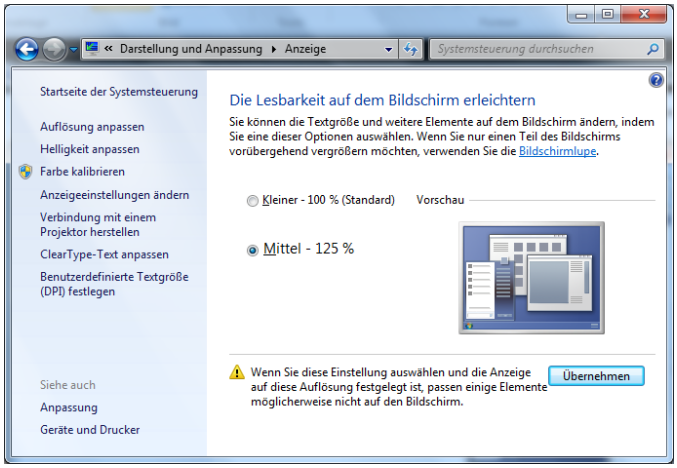

Abbildung 9: Die Windows-Einstellung „Mittel – 125%“

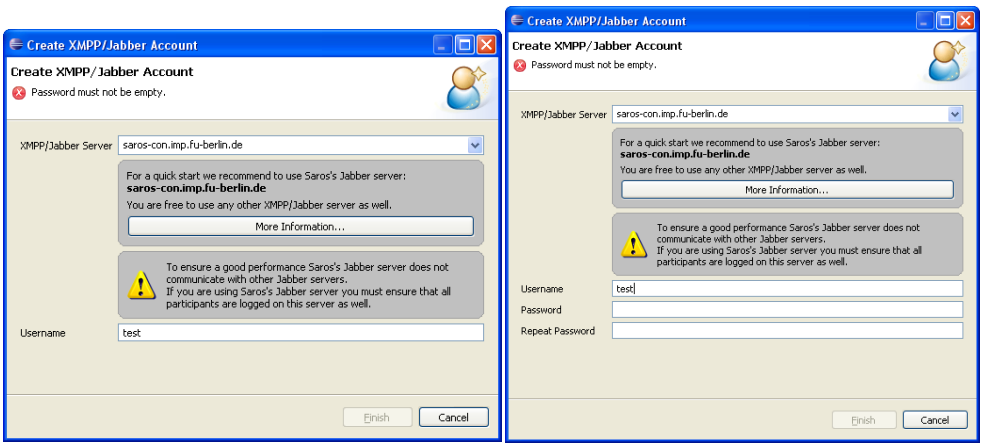

Abbildung 10: Links die abgeschnittene Darstellung, Rechts die korrekte Darstellung des Wizards

Nachdem die Nutzer es geschafft hatten, einen XMPP-Account zu konfigurieren, versuchten Sie sich gegenseitig als „Buddy“ hinzuzufügen. Ein Nutzer hatte hierbei in dem Dialog zum Hinzufügen eines Buddies, Schwierigkeiten zu verstehen, wie er dessen XMPP ID eingeben sollte, da diese vom Format her von den Beispielen im Dialog abwich, und die Bezeichnung der Eingabefelder für ihn nicht eindeutig war.

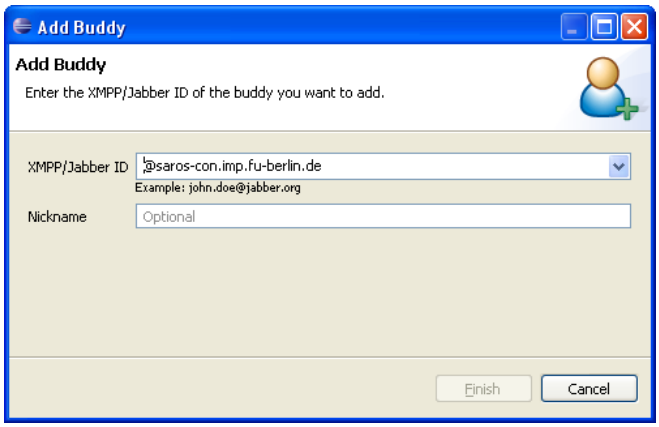

Abbildung 11: Initiale Darstellung des Dialogs zum Hinzufügen eines Buddies

Die tatsächliche XMPP ID des Nutzers war in diesem Fall „sven@awaldmann“, da der Nutzer sich mit einem lokalen XMPP-Server innerhalb des Firmennetzwerkes verbinden musste, da Saros Probleme beim Einsatz hinter der Firewall bzw. dem Unternehmens-Proxyserver hatte und eine Verwendung des offiziellen Saros-Servers somit nicht möglich war.

Der Nutzer gab nun als XMPP/Jabber ID „awaldmann“, den Servernamen, und als „Nickname“ den Namensteil der XMPP ID „sven“ ein. Die daraufhin erscheinende Fehlermeldung verwirrte ihn zunächst, allerdings erkannte er seinen Bedienfehler und er änderte seine Eingabe. Die Beispielangabe „john.doe@jabber.org“ war in diesem Fall nicht hilfreich, da seine XMPP ID etwas anders aufgebaut war.

Dies ist eines von mehreren Dialogelementen in Saros mit ungünstiger Benennung der Eingabeelemente, die eine Verzögerung bei der Bedienung des Systems zur Folge hatten. „Nickname“ meint in diesem Dialog eigentlich einen alternativen

Anzeigenamen für den Buddy innerhalb von Saros, damit Nutzer ihre Buddies schneller identifizieren können, und nicht eine lange XMPP ID (Beispiel: „waldmann@saros-con.imp.fu-berlin.de“) „lesen“ müssen.

Ein weiteres Problem, dass entdeckt wurde, ist die seltene und inkonsistente Nutzung der konfigurierten Anzeigenamen für Buddies. Diverse Dialoge, Fortschrittsbalken und Wizards ignorieren die Konfiguration des Anzeigenamens und stellen stattdessen die XMPP ID dar, sodass Nutzer den Sinn der Einstellung hinterfragten.

Weitere Probleme im Einrichtungsprozess betreffen unter anderem die von Nutzern als „penetrant“ empfundene Darstellung von Hinweisen, in Bezug auf eingeschränkte Nutzung des öffentlichen Saros-Servers „Saros-Con“, der von der Freien Universität Berlin kostenfrei bereitgestellt wird. Dieser XMPP Server ist so konfiguriert, dass Nutzer mit einem XMPP Account auf diesem Server nur mit anderen Nutzern kommunizieren und arbeiten können, die auch bei demselben Saros-Server angemeldet sind. Eine Kombination von jabber.org-XMPP IDs und Saros-Con-XMPP-IDs ist so beispielsweise nicht möglich.

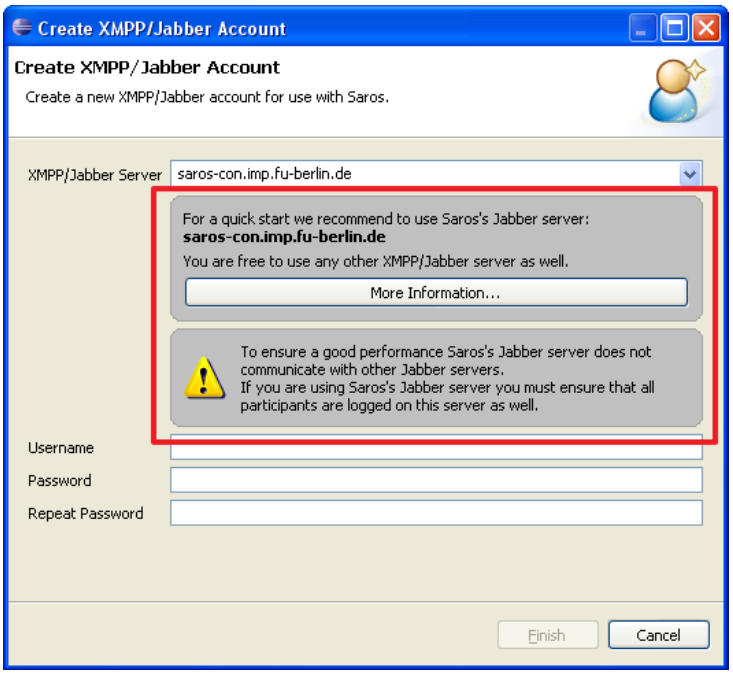

Abbildung 12: Hinweise auf Eigenheiten des öffentlichen Saros-Servers

Diese Hinweistexte wurden im Rahmen der Verbesserung der Out-of-Box-Experience in [Kahlert2011] hinzugefügt, da die Anwender durch fehlende Informationen über diese Einschränkung entsprechende Saros-Sitzungen versucht hatten, womit eine Zeitverschwendung einherging. Nutzer waren hiervon verwirrt, und lasen die Hinweise langsam durch, da ihnen nicht klar war, dass diese Hinweise sie nicht betreffen, da ein anderer Server zum Einsatz kommt.

Laut Dix et Al. (2003) sollten Dialoge keine Informationen beinhalten, die irrelevant sind oder selten benötigt werden,. Für diesen Dialog würde es sich somit anbieten, ihn auf mehrere Wizard-Seiten aufzuteilen, mit einer vorgeschalteten Auswahl, die

verhindert, dass ein Nutzer Hinweise sieht, die nur den Saros-Server betreffen, und bei Verwendung anderer Server somit irrelevant sind.

Weitere Beispiele für merkwürdige Meldungen in Saros, die bei einer Analyse der Usability unbedingt auffallen mussten, sind z.B. die Dialoge zum Hinzufügen eines Kontaktes, oder dem Entfernen eines Kontaktes, die einen groben Verstoß gegen die Heuristik N2. Übereinstimmung zwischen System und realer Welt darstellt:

> Das System sollte die Sprache des Benutzers sprechen. Dafür soll es keine systemorientierten Begriffe, sondern Wörter und Formulierungen, die dem Nutzer geläufig sind, nutzen. [Nielsen1994]

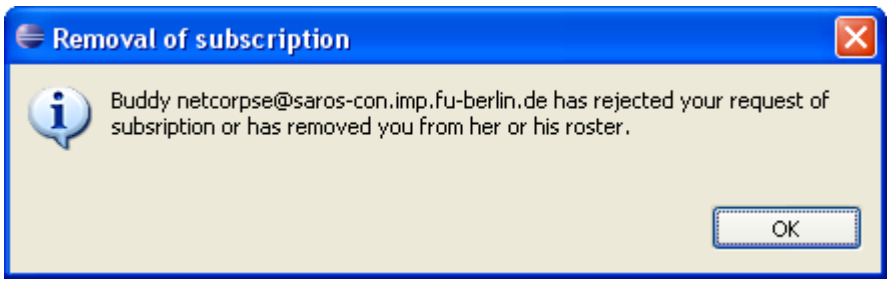

Abbildung 13: Technischer Dialog mit Rechtschreibfehlern und Fachbegriffen

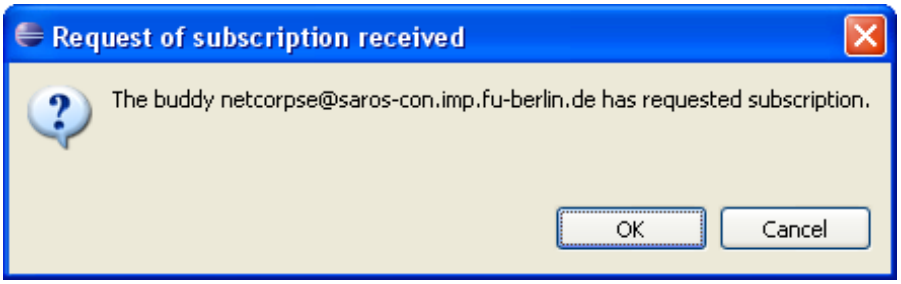

Abbildung 14: Technischer Dialog für das Bestätigen des Hinzufügens eines Buddies

Saros basiert größtenteils auf dem XMPP-Standard, weshalb sich in der Oberfläche noch viele Begriffe finden, die für Nutzer zunächst völlig unklar sind.

4.2.2 Probleme im Einladungsprozess

Der Einladungsprozess stellte bei Saros das größte Problem für die subjektive Zufriedenheit der Anwender dar. Der Einladungsprozess wurde im Rahmen der Arbeit von Solovjev [Solovjev2011] bereits als „wenig zusammenhängend“ beschrieben, und diese Einschätzung bestätigte sich in jeder Hinsicht: bei eigener Verwendung von Saros, bei der heuristischen Analyse und in den Usability Tests.

Da die Überarbeitung des Einladungsprozesses, neben einer Verbesserung der Awarenessinformationen, der Hauptfokus bei der Überarbeitung der Usability war, insbesondere, weil hier nicht nur die größten Probleme vorkamen, sondern auch das größte Potential für eine Verbesserung gesehen wurde, ist dieser Abschnitt detailliert.

Der Einladungsprozess besteht aus folgenden Schritten (unter Voraussetzung eines laufenden Eclipse mit erfolgreich hergestellter Verbindung zum XMPP Server):

Alice (künftig Host der Sitzung)		**Bob (der eingeladene Nutzer)**
0: Wizard öffnen 1: Auswahl von Buddies 2: Auswahl von Ressourcen 3: Checksummenberechnung 4: Versenden der Einladung mit kompletter Dateiliste in XML .. (warten) .. 5: Senden der Dateilisten mit Checksummen .. (warten) .. 6:Komprimieren der zu sendenden Dateien 7: Senden der ZIP-Datei	➔ 🡐 ➔ 🡐 ➔	.. (warten) .. .. 1: Empfangen der Einladung 2: Bestätigen der Einladung 3: Empfangen der Dateilisten 4: Anzeige des Wizards zur Ablage der ankommenden Projekte 5: Schließen des Wizards 6: Berechnung von Diffs 7: Senden der Diffs (Dateiliste) .. .. (warten) .. 8: Empfangen der ZIP-Datei 9: Entpacken der ZIP-Datei

Abbildung 15: Ablauf eines Einladungsprozesses

Die Reihenfolge der Interaktion von Alice kann hierbei anders ausfallen. So kann entweder die Wahl eines Buddies, oder die Wahl einer Ressource (z.B. einem Projekt) über die jeweiligen Kontextmenüs erfolgen. In diesem Fall verkürzt sich der Prozess da die Anzeige des Wizards entfällt.

Bereits bei Schritt 0 (Wizard öffnen) gibt es erste Probleme. Öffnet ein Nutzer den Wizard zum Starten einer Sitzung und liegt dabei eine Selektion in einem Navigator-artigen View vor, so wird diese Selektion übernommen. Dies erfolgt bei Klick auf einen der beiden folgenden Kontextmenüeinträge oder die Nutzung des Saros-Menüs.

Nach dem Klick waren Nutzer nicht sicher, ob sie tatsächlich den Menüpunkt angeklickt hatten, oder versehentlich daneben geklickt haben, da sich das Kontextmenü sofort schloss, aber erst mal (sichtbar) nichts passierte, da das System nun wie folgt vorging:

- Projektstruktur aus Workspace auslesen (ca. 3-5 Sekunden)
- Baumstruktur aufbauen (ca. 2-3 Sekunden)
- Selektion anwenden (Dauer: bis 10 Sekunden oder mehr)
- Wizard anzeigen

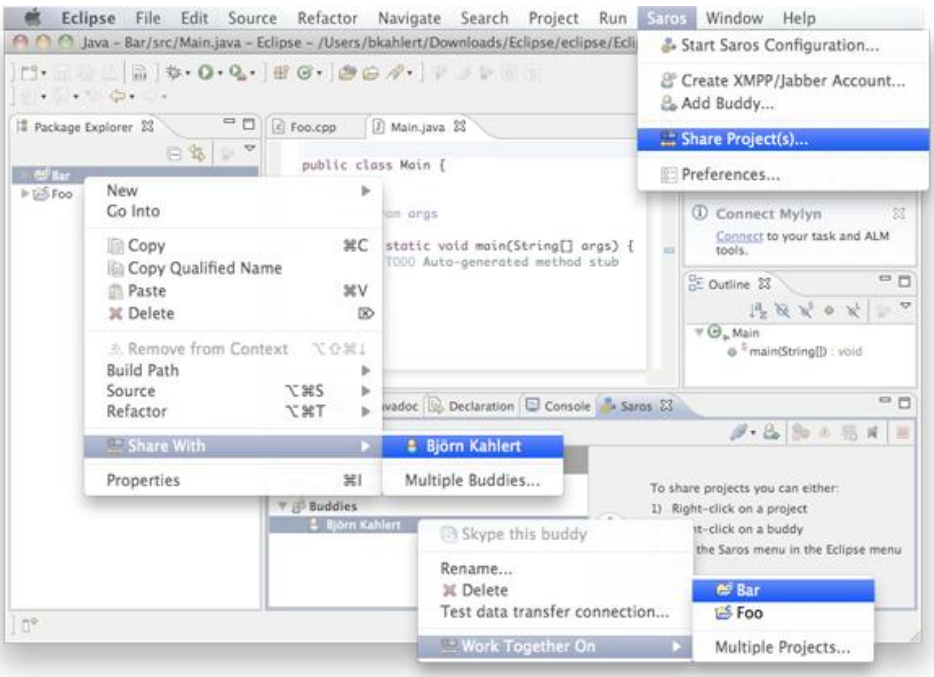

Abbildung 16: Menüs zum Starten einer Sitzung, Quelle www.saros-project.org

Da die Übernahme der Selektion blockierend implementiert wurde, und dies durchaus länger als 10 Sekunden dauerte, gab es für den Nutzer zunächst keine Rückmeldungen, abgesehen von einem „nervös" flackernden Mauszeiger.

Im darauffolgenden, verzögert angezeigten, Wizard zur Projektselektion, gibt es gleich mehrere Probleme.

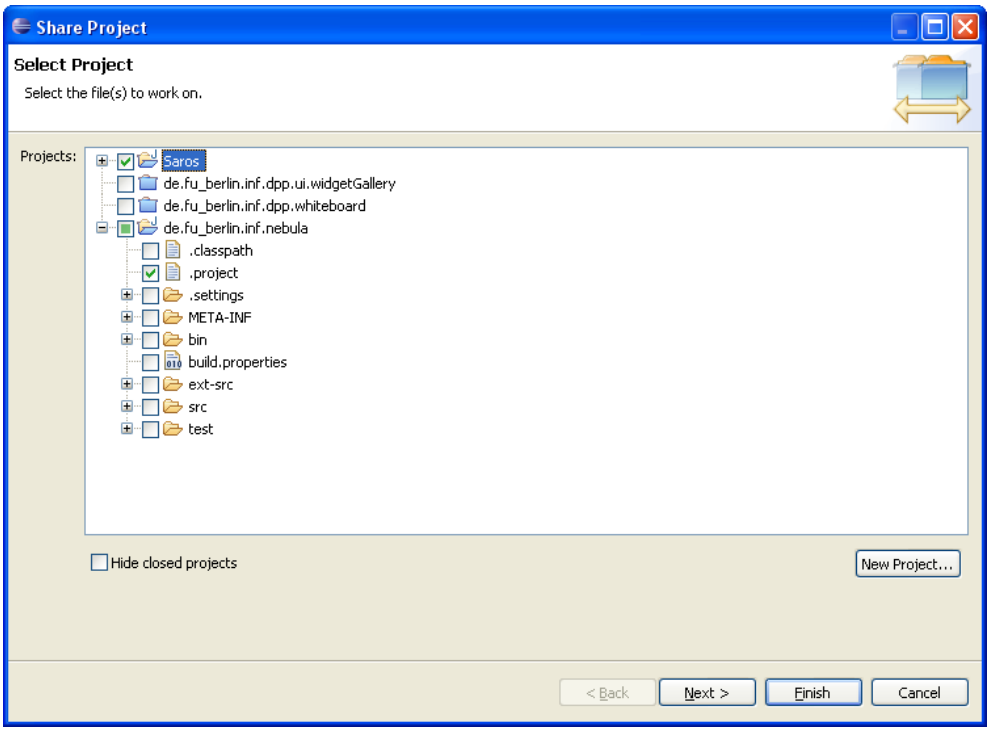

Abbildung 17: Die Wizard-Seite zum Auswählen von Ressourcen für die Sitzung

Die Sortierung der Ressourcen ist abweichend von der Standardsortierung in Eclipse: die Ressourcen werden ausschließlich alphabetisch sortiert. Dies ergibt eine ungeordnet wirkende Folge aus Dateien und Ordnern, wodurch die Orientierung erschwert wird. Auch die Beschriftungen in dem Wizard sind verwirrend: es ist unklar, ob nur ein Projekt, oder mehrere, oder sogar einzelne Dateien ausgewählt werden können. Nutzer verwendeten vermutlich dadurch die „Partial Sharing" Funktion erst, als sie nach den ersten Sitzungen mit Saros darauf hingewiesen wurden, dass es möglich ist, auch nur einzelne Dateien zu synchronisieren, und nicht nur Projekte gewählt werden können. Den Nutzern hat sich diese Funktionalität auch in mehreren Sitzungen nicht von alleine erschlossen.

Hat der Nutzer sich in diesem Dialog bei der Selektion der Ressourcen verklickt (beispielsweise nach komplexer Abwahl bestimmter Dateien im Dateibaum) so gab es keine Möglichkeit, die letzte Selektion rückgängig zu machen, und es musste von vorne begonnen werden (es gibt keine Schaltfläche für das Rückgängig machen und Wiederholen von Selektionen, auch Tastenkombinationen funktionieren nicht), was zu einer weiteren Verzögerung im Einladungsprozess führt.

Im nachfolgenden Schritt des Einladungsprozesses muss der künftige Host der Sitzung auswählen, welche Buddies zur Sitzung eingeladen werden sollen. Hierbei werden dem Nutzer irrelevante Optionen angeboten, deren Selektion zu Fehlermeldungen im Wizard führen würde. In der Liste der einzuladenden Teilnehmer im Einladungs-Wizard wurden alle Teilnehmer angezeigt, die der Nutzer zuvor als Buddy hinzugefügt hatte. Hier wurden auch solche angezeigt, die gerade online sind, ohne Saros zu nutzen (z.B. über ein InstantMessaging Programm) und solche, die gerade offline sind (nicht verbunden). Dies ermöglichte es Nutzern, Teilnehmer auszuwählen, die sie gar nicht erfolgreich einladen können, da dann eine Fehlermeldung im Wizard erscheint und sie nicht fortfahren können.

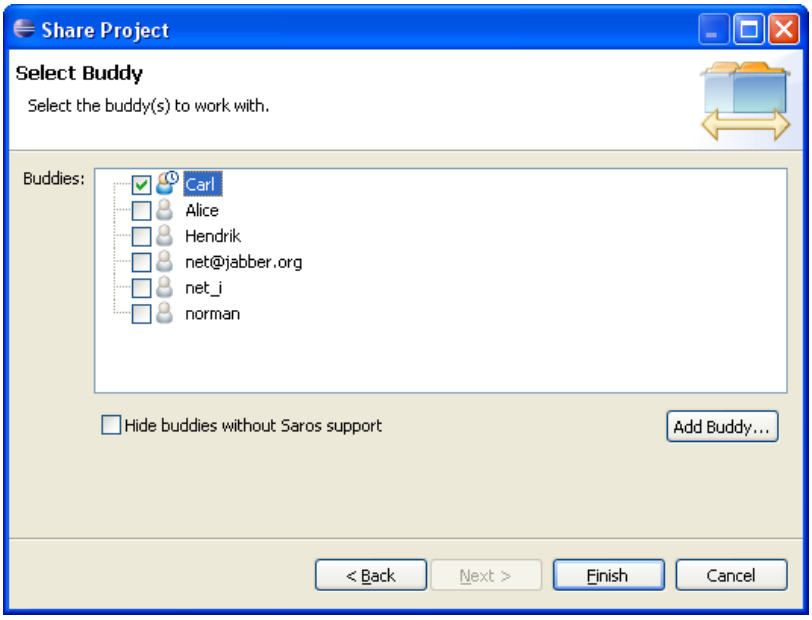

Abbildung 18: Die Auswahl der Sitzungsteilnehmer

Hatten Nutzer diese beiden Wizard-Seiten erfolgreich überwunden und den Wizard mit der „Finish“ Schaltfläche geschlossen, teilten sie den eingeladenen Nutzern mit, dass sie sie nun eingeladen haben. Diese sahen aber noch lange nichts, da das System nun voreilig (ohne, dass diese Informationen für die Sitzungseinladung an dieser Stelle benötigt wurde) die Checksummen für alle Dateien berechnete. Da hierzu zunächst der Projektzustand innerhalb von Eclipse aktualisiert werden musste, um mögliche Dateiänderungen auf der Festplatte, die Eclipse noch nicht bemerkt hatte, zu erkennen, dauerte dies besonders bei Festplatten mit beweglichen Teilen (HDDs) deutlich länger als 10 Sekunden (Laufzeiten bis zu einer Minute konnten beobachtet werden).

Während dieses Prozesses gab das System kaum Feedback. Durch die logische Trennung von Sitzungseinladung und dem Hinzufügen von Projekten zu einer Sitzung (eine technische Eigenheit, die sich den Nutzern nicht erschlossen hat) befindet sich der Host sofort in einer leeren Sitzung (alleine), was die einzige visuelle Veränderung in seiner Eclipse-Ansicht darstellt.

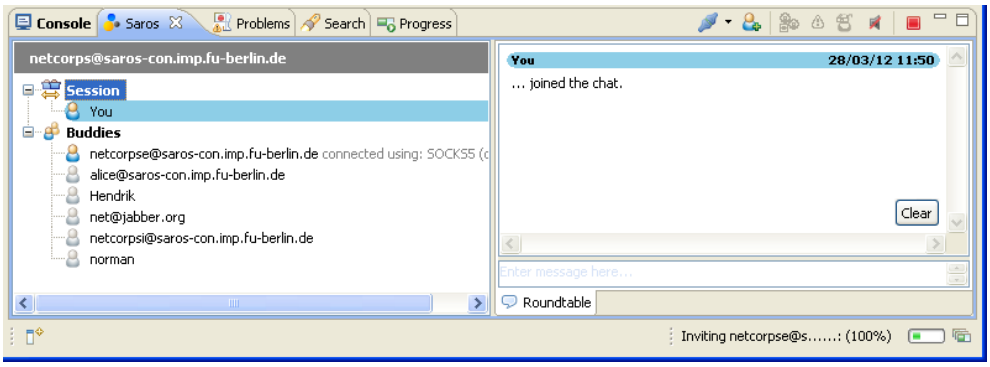

Abbildung 19: Die Ansicht für den Host nach Beenden des Wizards

Da der Nutzer kein Feedback darüber bekommt, ob die Einladung geklappt hat, wurde nun zumeist der eingeladene Nutzer gefragt, ob dieser etwas „bekommen hat“, was verneint werden musste. Daraufhin vermuteten einige Nutzer, dass es einen Fehler gab und die Einladung fehlgeschlagen ist und versuchten erneut über das Kontextmenü bzw. den Wizard Projekte zur Sitzung hinzuzufügen. Die Projekte, die für Saros bereits Bestandteil der Sitzung waren, konnten im Wizard nicht erneut ausgewählt werden, bzw. das erneute Einladen mit den Ressourcen hatte keinen Effekt.

Wenn nun die Checksummenberechnung nach langer Laufzeit (teilweise mehrere Minuten) abgeschlossen war, wurde die Einladung versandt und der eingeladene Nutzer sah einen Dialog mit einer Liste der eingehenden Projekte, der mehrere Schwächen aufweist:

- Er gibt keine Information, wieso diese Einladung versandt wurde
    - Nutzer müssen sich außerhalb von Saros koordinieren
- Er enthält keine Angaben von Dateigrößen
    - Nutzer wissen somit nicht, ob der Festplatten-Speicherplatz ausreicht um die Projekte zu empfangen (Rückfrage nötig)
- Er verwendet die JID und nicht den möglicherweise konfigurierten Alias
    - Nutzer hinterfragten den Nutzen der Umbenennen-Funktion, da der Alias nicht konsequent benutzt wird

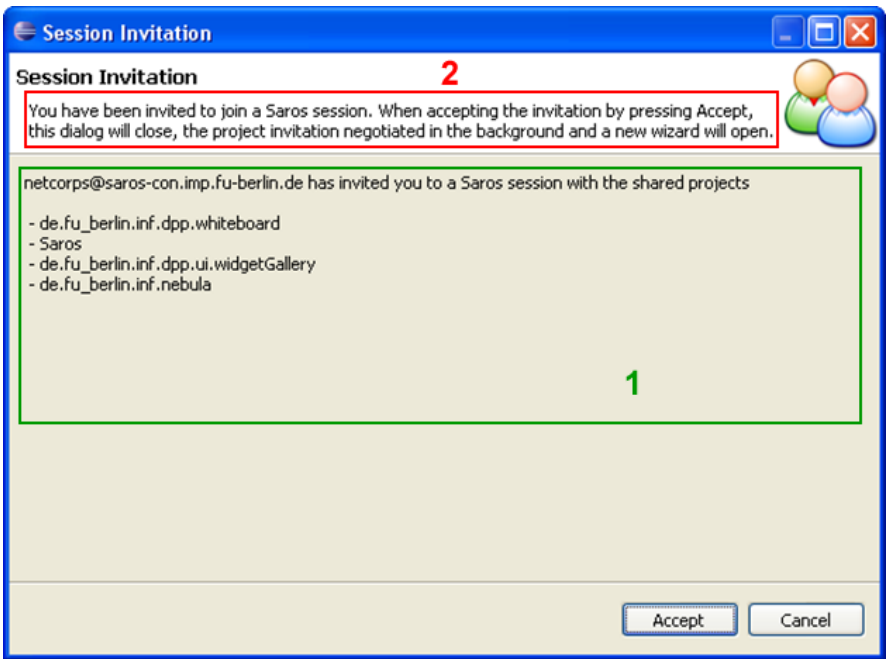

Abbildung 20: Einladung zu einer Sitzung. (1) Text der Einladung, (2) Hinweistext in der Wizard-Kopfzeile

Nutzern kommentierten in diesem Schritt des Einladungsprozesses den Text negativ, der oben (2) im Wizard darauf hinweist, das sich ein weiterer Wizard öffnen wird. Da bei Klick auf „Accept" erneut eine große Verzögerung auftrat, wurde diese Aussage von den Nutzern als „falsch" angesehen.

Erwähnenswert ist hier, dass der gesamte Text, der mit der JID beginnt und mit den Projektnamen endet, in der Saros-Instanz des einladenden Nutzers generiert und als „Einladungstext" versandt wird. Somit war es nicht möglich, hier den Alias anzuzeigen, da die Saros Instanz die den Text erstellt diesen nicht kennen kann. Bei der Korrektur dieser Implementierung stellte sich dann schnell heraus, wieso dies so gelöst worden war: es war technisch sehr umständlich, den Alias zu ermitteln. Dies ist vermutlich einer der Hauptgründe für das seltene Verwenden des Nutzer-Alias, da für das Auslesen dieser Eigenschaft, Referenzen auf unzählige Objekte in Saros nötig sind.

4.2.3 Probleme in der laufenden Sitzung

Hatten die Nutzer den Einladungsprozess überstanden, hatten sie in der laufenden Sitzung vor allem Probleme mit mangelnder Awareness. Es fehlten Hinweise über die Aktivitäten anderer Sitzungsteilnehmer und welche Ressourcen Bestandteil der laufenden Sitzung sind.

Durch die fehlende Kenntnis der Nutzer darüber, welche View-Arten von Saros übermittelt werden, und welche nicht, wurde mehrere Minuten lang erkundet, ob die Datei z.B. auf eine andere Art und Weise geöffnet werden muss, damit sie übertragen wird. Allerdings begriffen einige Probanden schon nach den ersten Sitzungen, dass bestimmte Editoren nicht übertragen werden können, und andere einwandfrei funktionieren und merkten sich dies, weshalb das Problem nicht in jedem Testlauf beobachtet werden konnte.

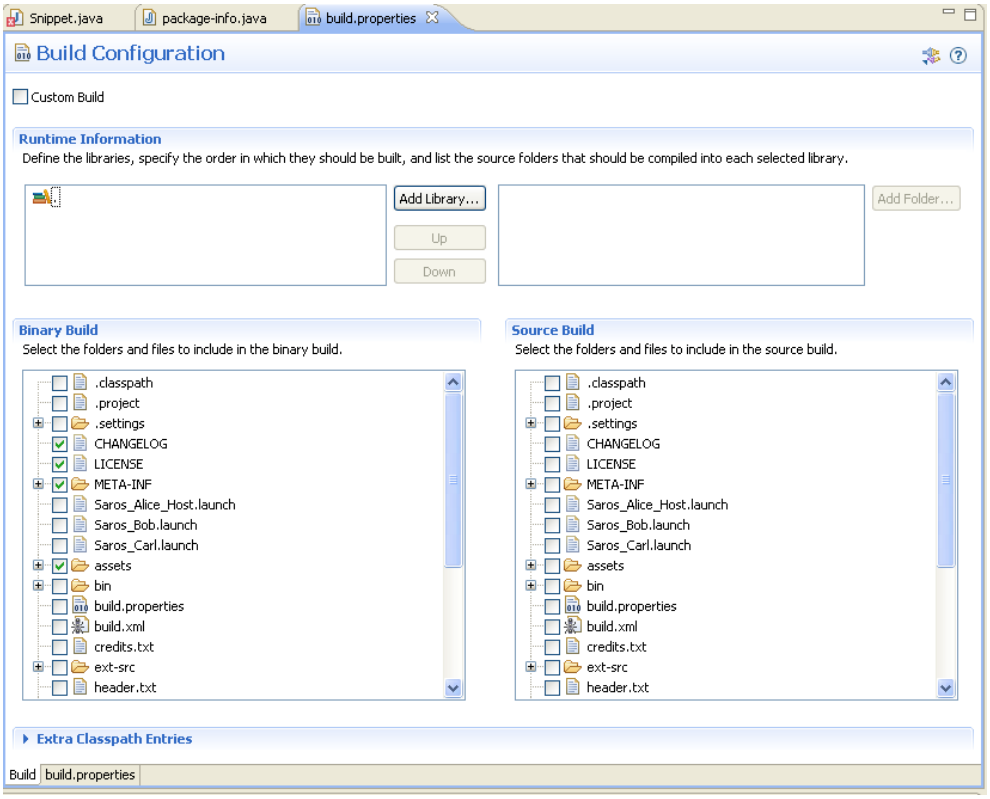

Abbildung 21: Editoren die aus mehreren Teilen bestehen, werden nicht synchronisiert

Das ein Editor-View nicht synchronisiert werden kann, wird von Saros nicht ausreichend kommuniziert. Befindet man sich im Follow-Modus und der verfolgte Nutzer öffnet eine solche Datei (z.B. die build.xml oder plugin.xml Dateien) erscheint eine kleine Ballon-Nachricht mit einem missverständlichen Hinweistext.

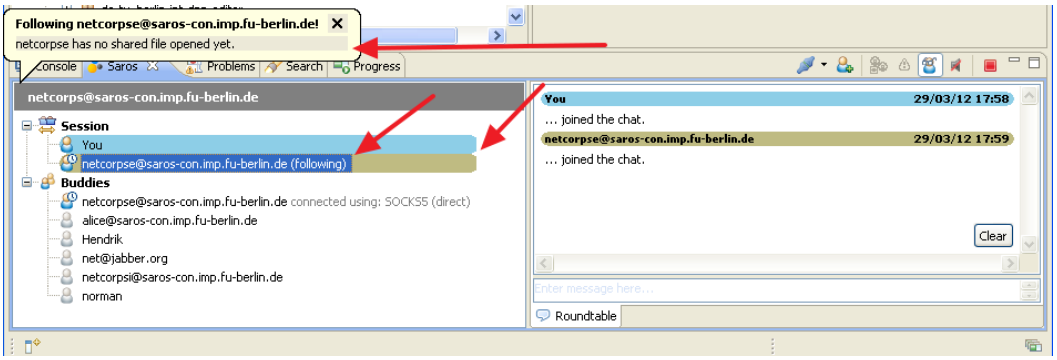

Abbildung 22: Benachrichtigung wenn eine Datei geöffnet ist im Follow-Modus

Die erscheinende Meldung „[User] has no share file opened yet“ ist missverständlich, da die Datei Sitzungsbestandteil ist, und lediglich der Editor-View nicht von Saros synchronisiert werden kann. Die Meldung suggeriert zunächst, dass die Datei nicht Bestandteil der Sitzung ist. Ohne den aktivierten Follow-Modus sieht der Nutzer in dieser Situation keinerlei Information darüber, was der andere Nutzer geöffnet hat.

Weitere mit Pfeilen markierte Probleme im Screenshot, sind ästhetischer Natur, sowie Probleme in der Handhabung:

- Die Farbe der Nutzer für die Sitzung wird als Hintergrundfarbe verwendet, dabei ist die Darstellung an einer Seite abgerundet, an der anderen kantig und je nach Bildschirmauflösung verpixelt
- Die Markierung des Baumelements stimmt nicht überein mit der farblichen Hinterlegung (Standard Hinterlegungs-Farbe)
- Der Text der Ballon Nachricht hat einen unpassenden grauen Hintergrund

Im folgenden Screenshot ist nun eine Nahaufnahme des Package Explorers während einer laufenden Saros-Sitzung zu sehen. Durch unzählige Markierungen an den Icons ist schwer zu identifizieren, welche Markierungen von Saros stammen und es ist unklar, welche Markierung welche Bedeutung hat.

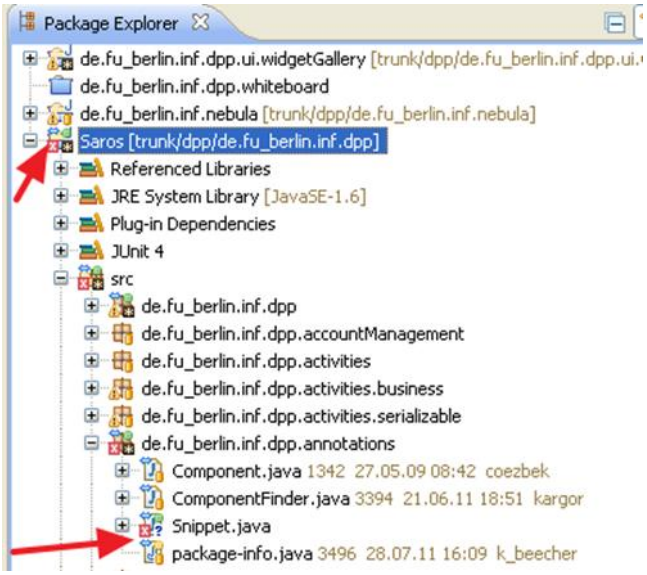

Abbildung 23: Der Package Explorer mit Icon-Markierungen

An Dateien die Bestandteil der Sitzung sind, erscheint ein kleiner blauer Doppelpfeil. Diese Markierung ist unzuverlässig, da die Darstellung des Baums in Eclipse zu selten aktualisiert wird. Somit erscheinen für den Nutzer an diversen Packages in diesem Screenshot keine Markierungen, auch wenn das komplette Projekt Bestandteil der Sitzung ist (dieses Problem löst sich, sobald Nutzer in der Sitzung eine Datei verändert haben).

Die kleinen grünen Kreise sollen andeuten, dass der Nutzer die Datei offen hat. Ein kleiner gelber Kreis zeigt, dass ein Nutzer die Datei im Hintergrund offen hat (d.h. Hintergrundtabs in Eclipse).

Bei dieser Markierung wäre es denkbar, dass die Farben der Nutzer verwendet werden für die Icon-Markierungen, was allerdings nur bei 2 Nutzern noch praktikabel wäre. Es gäbe mit dieser Methode schon keine sinnvolle Lösung über Icon-Markierungen, wenn mehr als 2 Teilnehmer dieselbe Datei offen haben.

So kam es in fast jeder Sitzung zunächst zu Verwirrung darüber, warum eine geöffnete Datei von Saros nicht synchronisiert wurde, und welche Projekte nun Bestandteil der Sitzung sind. Es erfolgte zu Beginn der meisten Sitzung eine Koordinationsphase, in der durch die Probanden experimentiert wurde, ob die Synchronisation klappt, bevor mit der eigentlichen Arbeit begonnen wurde.

Das am häufigsten auftretende Problem in Sitzungen, war die wiederholte Notwendigkeit verbaler Übermittlung von Präsenzinformationen zwischen den Probanden. Die häufigsten Fragen waren dabei:

- Was machst du gerade?
- Wo bist du gerade? (Bezogen auf Eclipse-Views / offene Dateien)
- Wer hat welche Datei offen?
- Wer ist der Host der Sitzung?
- Wer hat Schreibrechte?
- Wer folgt wem?

Die existierenden Mechanismen zur Darstellung, welche Datei ein Nutzer gerade offen hat, genügten nicht, um diese Information den Nutzern zugänglich zu machen. Bereits durch die heuristische Evaluation und die Ergebnisse aus [Solovjev2011] war klar, dass hier die größte Schwachstelle für die Nutzung von Saros liegt, daher wurde dieses Problem auch mit höchster Priorität behandelt, da hier signifikante Beeinträchtigung des Arbeitsflusses beobachtet wurde.

Einer der Mechanismen zur Darstellung von Awarenessinformationen, den Ballon-Benachrichtigungen, erwies sich als besonders problematisch. Bei der Betrachtung der Ballon-Nachrichten, wie in folgendem Screenshot zu sehen, ergaben sich schon in der heuristischen Evaluation unzählige Probleme, die auch in den Nutzertests bestätigt wurden:

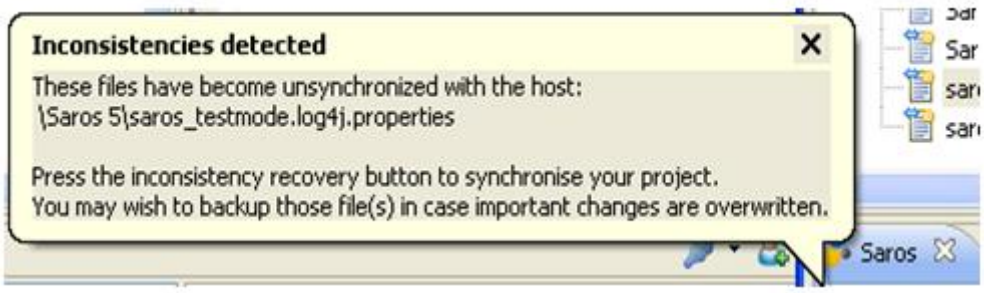

Abbildung 24: Mangelnde Ästhetik und Handhabung der Ballon-Benachrichtigungen

Ballon-Nachrichten lassen sich nur über das kleine X-Icon schließen, welches schwierig zu treffen ist. Zudem bleiben selbst kurze Benachrichtigungen 15 Sekunden lang bestehen. Mehrere Ballon-Nachrichten können sich überlappen und müssen alle einzelne geschlossen werden. Ästhetisch ansprechend wirkten diese auf die Probanden, aufgrund der unpassenden grauen Hintergrundfarbe des Textes, ebenfalls nicht. Für Windows-Nutzer weichen diese Benachrichtigungen in ihrer Handhabung von der Erwartungshaltung der Nutzer ab, die diese Art von Benachrichtigung vor allem von der Windows Menüleiste kennen.

Ein weiteres Problem ergab sich in der Handhabung von Inkonsistenzen, wie im oberen Screenshot zu sehen ist. Wenn eine Datei z.B. außerhalb von Eclipse verändert wird, reagiert die sogenannte „Watchdog"-Komponente von Saros, und informiert den Nutzer darüber, dass die lokale Version der Datei nicht mit der Version des Hosts der Sitzung übereinstimmt.

Diese Meldung wird auf dieselbe Art und Weise dargestellt, wie weniger wichtige Informationen, z.B. dass ein Nutzer noch keine Datei offen hat, oder der Follow-Modus beendet wurde. Dies wiederum führte dazu, dass die Nutzer die Benachrichtigung entweder lasen und dann ignorierten, oder diese sofort schlossen.

Nutzer werden hier nur darauf hingewiesen, dass sie ein Icon anklicken können (nicht müssen) und gleichzeitig wird auf die mögliche Notwendigkeit von Sicherungskopien hingewiesen (ohne Mitteilung wie dies vorzunehmen wäre).

Der Nutzer sieht sich mit diversen Fragen konfrontiert die eine Entscheidung schwer machen:

- Sollte eine Sicherungskopie angelegt werden?
- Wieso ist überhaupt die Inkonsistenz aufgetreten?
- Was passiert bei der Behebung der Inkonsistenz?
- Was passiert, wenn der Nutzer nicht auf das Icon klickt?

Die Nutzer wählten hier, bis auf eine Ausnahme, die letzte Variante: Sie ignorierten die Meldung und arbeiteten weiter, möglicherweise weil sie die Schaltfläche zum Behandeln der Inkonsistenz nicht fanden oder sich nicht gezwungen sahen diese zu suchen.

Nach dieser allgemeinen Vorstellung zentraler Probleme in Saros, folgen im nächsten Abschnitt Einblicke in das mentale Modell der Probanden, und deren subjektive Zufriedenheit mit Saros.

4.2.4 Mentales Modell der Probanden

In einigen Situationen kam es dazu, dass die Probanden ihre Interpretation der Systembegriffe und der Konzepte in Saros aussprachen, was Hinweise auf problematische Bezeichnungen und ungünstiges Systemverhalten lieferte.

So kam es, durch die große Verzögerung nach dem Senden einer Einladung, bevor Saros einen Fortschrittsbalken anzeigt, bei einem Probanden dazu, dass dieser anfing, nach einer gestarteten Einladung, alle gerade offenen Editor-Fenster in Eclipse zu schließen. Er vermutete, dass dies den Start des Einladungsprozesses verhindert. Genau in dem Moment, als er den letzten Editor geschlossen hatte, erschien ein Fortschrittsbalken für das Versenden der Einladung an einen Teilnehmer. Der Nutzer war fortan der festen Überzeugung, dass er für den erfolgreichen Start des Einladungsprozesses zunächst alle offenen Editoren schließen müsse, auch wenn dies nicht der Fall war.

Die Ursache für dieses Problem kann recht eindeutig darauf zurückgeführt werden, dass Saros nicht sofort Rückmeldung darüber gibt, dass die Einladung bereits gestartet wurde. Nach dem dies behoben wurde, konnte das Problem in den danach durchgeführten Tests nicht mehr beobachtet werden.

In einer anderen Sitzung (Paarprogrammierung zu zweit, mit Rollentausch durch Aufbau einer neuen Sitzung) äußerte ein Nutzer sich dahingehend, dass der Host „immer Pink ist“, und eine feste Farbe habe.

Einer der beiden Nutzer hatte in der ersten Sitzung seine ihm automatisch zugewiesene Farbe auf Pink umgestellt (ein häufig beobachtetes Verhalten der Probanden). Nach dem Wechsel der Rollen, dadurch dass der Nutzer der zuvor eingeladen hatte, nun vom anderen Nutzer zu einer Sitzung eingeladen wurde, hatte der einladende Nutzer plötzlich die Farbe des anderen Nutzers. Dies suggerierte offenbar dem Nutzer, dass der Host stets eine besondere Farbe erhält.

In einer weiteren Situation war ein Nutzer sehr verwundert, als der von ihm eingeladene Nutzer (Observer) in einer Sitzung eine Änderung der offenen Dateien vornehmen konnte, in der die beiden Teilnehmer explizit Driver und Observer Rollen für die Paarprogrammierung eingenommen hatten. Er äußerte sich bei der Diskussion mit seinem Kollegen über Saros dahingehend, dass doch „nur der Host schreiben darf".

Hier offenbarte sich eine Verwechselung der Rolle „Host" mit dem alleinigen Recht, schreiben zu dürfen. Dies kann darin begründet sein, das Saros keine Informationen darüber bereitstellt, wer in einer Sitzung Schreibrechte haben wird, nach der Einladung.

4.2.5 Ergebnisse der Interviews und sonstigen Befragungen

Nach den Usability-Tests wurden die Probanden nach ihrer Einschätzung von Saros gefragt, und danach ob sie Probleme hatten, und wenn ja, welche Probleme und wie schwerwiegend diese Probleme waren. Auch wurden sie befragt, welche neuen Funktionen sie sich wünschen würden.

Neben kurzen unstrukturierten Befragungen, direkt im Anschluss an die Tests wurden 15-30 Minuten dauernde Interviews mit den Probanden geführt, nachdem sowohl das unveränderte Saros, als auch die ersten Prototypen getestet wurden.

Hieraus ergab sich, dass die Nutzer grundsätzlich überzeugt waren von der Idee hinter Saros, und Saros gerne weiter einsetzen würden. Allerdings hatten sie stets Anmerkungen zu Problemen die bei der Benutzung auftraten, und vermuteten, dass Saros eine alpha-Version (sehr unreife Software) wäre.

Die Wünsche und Probleme der Probanden beinhalteten vor allem:

- Eine Übertragung des Webcam-Bildes, um z.B. zu sehen, ob der andere Nutzer konzentriert ist
- Übermittlung von mehr Awarenessinformationen
    - Keiner der Teilnehmer wusste ohne verbale Koordination, was die jeweils anderen Teilnehmer machten
- Farbbalken an den Seiten der Editoren sind nicht eindeutig genug
- Einladungsprozess ist langsam und gibt zu wenig Feedback

Die Probanden waren besonders frustriert durch fehlende Rückmeldungen über den Systemzustand und die fehlenden Informationen über geöffnete Dateien anderer Nutzer, sowie die mangelnde Indikation der Position des Cursors bzw. Mauszeigers. Die bestehenden Awareness-Mechanismen wurden als missverständlich und ungeeignet bezeichnet. Einer der Probanden äußerte sich

dahingehend, dass er „nie wusste was die anderen Teilnehmer tun“, andere Probanden waren in der Kritik zurückhaltender.

4.3 Reliabilität der Ergebnisse

Bei der Durchführung von Usability Tests ist es wichtig, sowohl die Reliabilität als auch die Validität der Ergebnisse zu berücksichtigen. Die Validität meint die Aussagekraft der Ergebnisse: Wurden tatsächlich Usability Probleme gefunden, die von Bedeutung sind für das getestete Produkt? Eine hohe Reliabilität hingegen bedeutet eine hohe Wahrscheinlichkeit, dass eine erneute Durchführung des Testlaufs gleiche Ergebnisse liefern würde (Vgl. [Sarodnick2006], [Nielsen1993]).

Für Tests der Eignung von Saros für verteilte Paarprogrammierung und Code-Reviews wurden die Aufgabenstellungen aus Abschnitt 4.1.3 bearbeitet. Für die Tests des unveränderten Saros, sowie der Prototypen, wurden hierbei unterschiedliche Programmieraufgaben vorgegeben. Aufgrund der geringen Anzahl an Probanden und unternehmensinternen Terminverschiebungen und Krankheitsfällen war es nicht möglich, die Probanden so aufzuteilen, dass für die Evaluation der Prototypen zwei andere Probanden dieselbe Aufgabenstellung bearbeiten, wie bei der Evaluation des unveränderten Saros-Release. Es ist somit wahrscheinlich, dass abweichende Ergebnisse bei anderer Kombination der Probanden oder anderer Wahl der Programmieraufgaben erzielt würden. Auch die individuellen Unterschiede zwischen den Probanden können große Auswirkungen haben. So kann der beste Nutzer bis zu 10-mal schneller arbeiten als der langsamste Nutzer. Die 25% der besten Nutzer sind meist doppelt so schnell wie die langsamsten 25% der Nutzer (Vgl. [Nielsen1993, S.166]). Da die Probanden feststehen und keine alternativen Probanden akquiriert werden konnten, muss dies für diese Arbeit akzeptiert werden.

Es war auch nicht möglich, drei der Probanden auf einmal in einer Sitzung zu beobachten. Da das Szenario mit drei Nutzern als weniger wichtig angesehen wurde, wurde es hier als akzeptabel angesehen, dass es davon nur Aufzeichnungen gibt, in denen ich einer der Teilnehmer war (teilnehmende Beobachtung). Dies könnte die Ergebnisse verfälschen, da ich zum Zeitpunkt der Tests bereits umfangreiches Wissen über die Implementierung von Saros hatte, und mich somit unrealistisch verhalten könnte. Allerdings sollte die Kenntnis des Codes nicht die Nutzung der Software ausschließen, weshalb die Erkenntnisse aus diesen Tests in die Auswertung einflossen.

Da nur 4 Probanden für die Tests zur Verfügung standen, die auch für die Evaluation der Prototypen zum Einsatz kamen, kann zudem die Reifung der Tester ein Problem darstellen. Die Reifung (Maturation) der Tester bezieht sich auf deren Erfahrung im Umgang mit der Software bei deren Anwendung. Dies kann Einfluss nehmen auf die Testergebnisse, da die Nutzer bei Kenntnis des Systems, ihnen bekannte Probleme bei späteren Tests leichter umgehen können. Um diesem Problem der Reifung zu entgehen bietet es sich an, für spätere Usability-Tests neue Probanden zu akquirieren (Vgl. [Nielsen1993]).

Da keine neuen Probanden akquiriert werden konnten, wurde zur Verringerung der möglichen Beeinflussung der Ergebnisse, darauf geachtet, dass die Evaluation

der Prototypen in anderen Konstellationen der Probanden erfolgte, als die Tests der unveränderten Version von Saros.

Da nur wenige Probanden zur Verfügung stehen, könnten die Eigenheiten der Probanden bedeutsam sein. Bei der Betrachtung der Eigenheiten fiel auf, dass sich einer der Probanden bei der Aufgabenbearbeitung meist in eine Observer- (eher eine Beobachter-) Rolle begab, und eher passiv agierte. Es ist unklar, ob dies der Fall war, weil andere Probanden in den Tests dominanter agierten, oder ob es an der grundlegenden Einstellung des Probanden lag. Es ist wahrscheinlich, dass sich bei einer genaueren Vorgabe einer Paarprogrammierungs-Aufgabe, mit vorgegebener Rollenverteilung, abweichende Ergebnisse ergeben hätten. Auch kann durch unterschiedliche Herangehensweisen der Probanden, in der Konzeptionsphase, bei der Lösung einer der vorgegebenen Programmieraufgaben, eine Wiederholung der Tests, in anderen Konstellationen der Probanden, unterschiedliche Ergebnisse ergeben.

Da es bei der Beobachtung der Probanden allerdings nicht darum ging, zu ermitteln, ob diese die Programmieraufgaben erfolgreich lösen, sondern zu ermitteln, welche Probleme es bei der Bedienung des dabei verwendeten Systems gibt, nehme ich hier an, dass es keine negativen Einflüsse auf die Ergebnisse geben wird.

4.4 Validität der Ergebnisse

Bei der Betrachtung der Validität der Tests ist es wichtig zu betrachten, ob die durchgeführten Usability-Tests etwas messen, was tatsächlich Bedeutung bei der Nutzung des Produkts im praktischen Einsatz, außerhalb etwaiger kontrollierter Laborumgebungen hat. Während die Reliabilität der Tests mit statistischen Tests geprüft werden kann, ist zum Erreichen einer hohen Validität Verständnis für das methodische Vorgehen, sowie der Einsatz gesunden Menschenverstandes notwendig [Nielsen1993, S.169].

Typische Validitätsprobleme wären die Wahl der falschen Nutzer, sowie die Vergabe unpassender Aufgaben. Auch zeitliche und soziale Einflüsse können negative Auswirkungen haben (Vgl. [Sarodnick2006], [Nielsen1993, S.169f]). Nielsen unterscheidet zwischen Anfänger- und Experten-Nutzern, wobei Anfänger-Nutzer potenziell Probleme mit allgemeinen Interaktionskonzepten (und der Hardware) eines Systems haben könnten, was negative Auswirkungen auf die Validität des Tests hätte und zunächst Schulungen der Benutzer erfordern würde [Nielsen1993, S.177]. Dies ist bei den durchgeführten Usability-Tests nicht der Fall, da die Probanden sich mit der Entwicklungsumgebung Eclipse auskennen und keinerlei Schwierigkeiten haben, ihren Arbeitscomputer mit Maus und Tastatur zu bedienen.

Die Wahl der Aufgabenstellungen erfolgte in Anlehnung an die Erkenntnisse aus der Saros-Nutzerumfrage. Sie entsprechen damit sowohl der tatsächlichen Nutzung von Saros (bis auf die Sitzungen mit mehr als drei Teilnehmern) als auch den Nutzungsszenarien für die Saros der Webseite zufolge gut geeignet sein soll.

Vor der Testdurchführung wurde den Probanden das System präsentiert und eine Einführung in die grundlegende Konzeption der Software gegeben, ohne hierbei

Funktionen wie VoIP und Screensharing zu erwähnen, da diese aufgrund des experimentellen Zustands und der hohen Anzahl an bekannten Fehlern nicht genutzt werden sollten. Zudem hatten sie die Möglichkeit sich eigenständig über den Funktionsumfang und technische Aspekte auf der Saros-Webseite zu informieren.

Für die Durchführung der Usability-Tests bzw. der Beobachtungen der Probanden bei der Nutzung des Systems gab es keinen zusammenhängenden Termin, an dem alle Probanden für Usability-Tests zur Verfügung standen. Stattdessen gab es diverse Termine über mehrere Wochen verteilt, in denen jeweils 2-3 der Probanden Zeit hatten, im Rahmen ihrer regulären Tätigkeit im Unternehmen Saros einzusetzen und dies aufzeichnen, sowie durch mich beobachten zu lassen.

Aufgrund der technischen und räumlichen Ressourcen im Unternehmen war es bei einigen der Sitzungen nicht möglich, sicherzustellen, dass die Probanden tatsächlich weit voneinander entfernt, in unterschiedlichen Räumen mit Saros arbeiten. Da allerdings aufgrund der Anordnung der Arbeitsplätze in dem jeweiligen Räumen keine Möglichkeit existierte, dass ein Proband den Bildschirminhalt eines anderen Probanden sehen konnte, und der jeweils andere Proband hinter seinen Monitoren „versteckt" war, gehe ich nicht einer Beeinträchtigung der Testergebnisse aus. Die Probanden wussten, dass es darum geht, eine Software für verteilte Programmierung zu testen, und als Beobachter wäre ein Eingreifen möglich gewesen, um Verhalten zu verhindern, das den Test verfälscht hätte. Zwar konnten die Probanden sich innerhalb des Raumes sprechen hören, allerdings hatten die Probanden in allen Sitzungen eine VoIP-Verbindung aufgebaut und über Headsets miteinander kommuniziert, weshalb keine negativen Auswirkungen erwartet wurden. Gerade durch diese Konstellationen stellte sich im Laufe der Tests heraus, dass die Probanden auch Gefallen daran fanden, Saros einzusetzen, wenn Sie sich nicht in unterschiedlichen Abteilungen, Räumen oder Gebäuden befanden, da die Anordnung der Tische und Arbeitsplätze teilweise keine bequeme Partnerarbeit vor demselben Rechner ermöglichte.

Bei der Betrachtung der Eignung der Probanden für die Tests fällt auf, dass diese zwar die notwendige Erfahrung im Umgang mit Eclipse und Paarprogrammierung haben, aber bei den Tests vier ausschließlich männliche Probanden zum Einsatz kamen. Dies liegt unter anderem daran, dass der Anteil der Frauen in der Informatik bedauerlich gering ist. Als Beispiel sei hier der Anteil der Frauen als Studierende der Informatik (und damit als zukünftige potenzielle Saros-Nutzer) angeführt. Der Anteil der Frauen in den Informatikstudiengängen an der Freien Universität ist mit 12% im Wintersemester 10/11 gering. [23]

[23] http://www.mi.fu-berlin.de/beauf-aussch/beauf-frauen/ressourcen/Massnahmenplan_FBMatheInfo-final.pdf

Zielvereinbarungen 2010/2011
Maßnahmenplan zur Frauenförderung, Gleichstellung und Familienfreundlichkeit
Fachbereich Mathematik und Informatik der Freien Universität Berlin
Seite 10 von 14

Anlage 4b: Studierende der Informatik (Diplom, Mono-BSc., MSc.)

	WS 00/01	WS 03/04	WS 07/08	WS 10/11
Anzahl gesamt	580	757	931	1122
weiblich	116	104	129	133
weiblich (%)	20	14	14	12

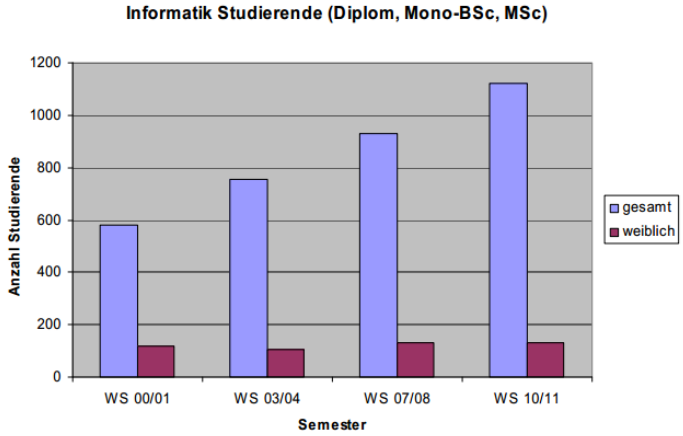

Abbildung 25: Anteil der Frauen in den Studiengängen der Informatik,

Quelle: Webseite der Freien Universität Berlin

Die Versuche, wenigstens eine weibliche Entwicklerin aus dem Unternehmen zu akquirieren, scheiterten daran, dass diese zeitlich nicht zur Verfügung standen. Der Anteil an Feedback von weiblichen Software-Entwicklerinnen stammt somit ausschließlich aus dem Feedback der beiden Teamkolleginnen (eine davon die Diplomarbeitsbetreuerin) und den gender-unspezifischen anonymisierten Rückmeldungen über die Nutzerumfragen.

Nach Faulkner (2003) besteht das Risiko einer geringen Quote an aufgedeckten Usability-Problemen durch zu wenig Teilnehmer oder eine nicht ausreichend verschiedene Teilnehmergruppe, was sich mit den Bedenken von Nielsen (1993) deckt. Da allerdings keine Erkenntnisse vorliegen, dass Frauen beim Softwareentwickeln anders agieren als Männer, oder andere Anforderungen an die Usability von Software haben, und es auch keine Hinweise gab, dass der Anteil an Frauen bei den Nutzern von Saros hoch wäre, sehe ich diese Verteilung der Geschlechter bei den Probanden als unproblematisch an. Auch in der Verteilung des Alters der Probanden sehe ich keine Probleme. Die Probanden sind zwischen 22 und 32 Jahren alt. Ich kann nicht ausschließen, dass ich dadurch Probleme nicht identifizieren werde, die nur bei älteren Anwendern von Saros auftreten würden. Da ich allerdings davon ausgehe, dass ich auch bei nur 4 ähnlichen Probanden, nicht in der Lage sein werde, alle gefundenen Probleme zu beheben, werden mir die Ergebnisse als Arbeitsgrundlage ausreichen.

Problematisch könnten allerdings die beruflichen Beziehungen zu den Probanden sein. Die Probanden sind meine Arbeitskollegen, aus demselben Team im Unternehmen. Dadurch, dass ich wesentlich mehr Erfahrung in der Softwareentwicklung habe, und schon lange in diesem Unternehmen arbeite, besteht eine Art „Senior zu Junior“ Verhältnis zu den Probanden.

Dies kann Einfluss nehmen auf den von den Probanden empfundenen Leistungsdruck. Ich bin zwar kein Vorgesetzter der Probanden, allerdings mindestens als eine Respektsperson anzusehen. Die Probanden wurden wiederholt daran erinnert, dass Ihre Leistung nicht bewertet wird, und sie nichts falsch machen können, da es nur darum geht, die benutzte Software zu testen. Dennoch war zu beobachten, dass die Probanden bestrebt waren, eine möglichst gute Leistung zu erbringen. Mögliche negative Folgen, die dies haben könnte, sind nach Nielsen (1993) sogar nervliche Zusammenbrüche der Probanden, oder Tränenausbrüche. Aufgrund meiner Einschätzung der Kollegen konnte ich aber ausgehen, dass dies nicht passiert und es keine Probleme geben wird bei der Durchführung der Tests.

Bei der Betrachtung möglicher Einflussnahme durch soziale Faktoren, sind die Probleme bei einer der Evaluierungen der Prototypen zu erwähnen. Bei der letzten Evaluation waren zwei Saros-Teamkollegen, die für die Probanden fremde Personen sind, bei den Tests mit anwesend. Die Probanden wirkten nervös und eher unkonzentriert bei der Aufgabenbearbeitung. Allerdings konnten keine negativen Auswirkungen auf die Analyse der Prototypen festgestellt werden, da die Probanden nach anfänglicher Zeitverzögerung, die Aufgaben zielorientiert bearbeiteten, auch wenn die Bearbeitungsdauer der Aufgaben deutlich länger war.

Bei einer abschließenden Betrachtung der Aussagekraft der Ergebnisse ist zu berücksichtigen, dass die Anzahl der befragten Nutzer gering ist, und quantitative Daten damit wenig aussagekräftig wären, weshalb hier kaum Daten erhoben wurden, da die dadurch eingesparte Zeit auf die Behebung der Probleme verwendet werden konnte. Das Problem einer zu geringen Aufdeckungsquote der Usability-Probleme wie sie von Faulkner (2003) erwähnt wird, liegt offensichtlich nicht vor, bei über 100 identifizierten Problemen, darunter vielen gravierenden.

Die in Abschnitt 5.3 folgenden Annahmen über die subjektive Zufriedenheit der Nutzer und die Verbesserung selbiger, müssen allerdings zurückhaltend erfolgen. Es ist möglich, dass die Rückmeldungen der Nutzer bei der Evaluation der Prototypen positiver ausgefallen sind, da diese in die Verbesserung der Usability involviert waren, was nach Nielsen (1993) Einfluss auf die subjektive Zufriedenheit der Nutzer nehmen kann.

Auch ist es möglich, dass die Nutzer sich über den Testverlauf so sehr in das System eingearbeitet haben, dass in späteren Evaluationen von Prototypen, einige Probleme nicht mehr beobachtet wurden, da die Probanden diesen (unbewusst) aus dem Weg gegangen sind. Um dies zu vermeiden, wären Tests mit mehr, bzw. mit anderen Probanden mit Langzeit-Erfahrung beim Einsatz von Saros notwendig gewesen, was für diese Arbeit nicht möglich war.

Zusammenfassend lässt sich festhalten, dass eine negative Beeinflussung der Reliabilität und Validität durch die Methodik und die Wahl der Probanden nicht vollständig ausgeschlossen werden kann. Präzise Aussagen über den Grad der erzielten Verbesserung sind aufgrund der geringen Anzahl der Probanden nicht möglich. Dennoch wurden die erzielten Ergebnisse als ausreichend angesehen, um sie als Grundlage für eine Überarbeitung der Usability von Saros zu

verwenden, die im folgenden Kapitel beschrieben wird, und die Ergebnisse aus den standardisierten Fragebögen wurden als geeignete Indikation für eine tatsächlich erzielte Verbesserung des Systems angesehen.

5 Verbesserung der Usability

Nach der exemplarischen Vorstellung einiger der beobachteten Problemen bei den Usability-Tests, wird in diesem Kapitel beschrieben, wie die Usability auf Basis des Problemkatalogs überarbeitet wurde, und welche Lösungskonzepte dabei entstanden sind.

Nach der Heuristischen Evaluation und den ersten Usability Tests wurden die größten, identifizierten Probleme, als abstrakte Ziele für die Überarbeitung von Saros formuliert, die bei der Zusammenstellung der Arbeitspakete als Leitlinie dienten:

1. Die Geschwindigkeit des Einladungsprozesses soll verbessert werden
2. Nutzer sollen stets wissen, was Saros tut, wenn es aktiv ist
3. Nutzer sollen den Einladungsprozess als schnell empfinden
4. Wichtige Präsenzinformationen sollen übermittelt werden,
   für eine gesteigerte Effizienz

Aus dem Problemkatalog wurden dazu anhand der Problemgewichtung und der Funktionsgruppe Arbeitspakete zusammengestellt, die dann bearbeitet wurden.

Ziel dieser Arbeit war es, größere Usability-Probleme im Einladungsprozess in Saros zu beseitigen (Probleme mit einer Gewichtung von 3 oder 4), und dabei auch die anderen Probleme zu behandeln, die in den bearbeiteten Komponenten auftreten. Dies liegt vor allem daran, dass es einen hohen Einarbeitungsaufwand in die jeweiligen Code-Bereiche erfordert, und die Behebung der kleineren Probleme somit vom Aufwand/Nutzen Verhältnis wieder attraktiv werden, da für diese die erfolgte Einarbeitung genutzt werden kann. Mindestens eine Komponente von Saros soll zukünftig auf den Nutzer „durchdacht“ wirken und nicht mehr wie eine Alpha-Version.

Der Fokus lag klar auf dem Einladungsprozess, sowie einer Beseitigung der beobachteten Mängel bei der Darstellung von Awareness-Informationen. Probleme aus den einzelnen Funktionsgruppen wurden in Arbeitspaketen zusammengefasst, die dann in Entwicklungs-Sprints von jeweils 1-2 Wochen bearbeitet wurden. Hierbei wurden Probleme mit hoher Gewichtung (von 3-4) bevorzugt gewählt, und dann um dazu passende kleinere Probleme ergänzt, die in demselben bearbeiteten Code-Teil vorlagen.

Bereits eine kleine textuelle Veränderung in der Oberfläche führte zu hohem Nachbesserungsbedarf, da hierdurch häufig automatisierte Tests von Saros fehlschlugen, weshalb keine der Änderungen als trivial angesehen werden konnte. Somit wurde ein großer Zeitpuffer eingeplant, für unerwartete Verzögerungen.

Einige Probleme warfen zudem Grundsatzfragen zur Struktur bzw. Architektur von Saros auf, und führten zu umfangreichen Diskussionen im Saros-Team, aber letztlich zum Entschluss, diese nicht zu bearbeiten. Darunter fällt insbesondere die Beseitigung der technischen „Host“-Rolle.

5.1 Arbeitspakete

In den folgenden Abschnitten werden nun die Arbeitspakete präsentiert, und die Motivation für die Veränderungen, in Form von Zielsetzungen, wird dargestellt.

5.1.1 Arbeitspaket: Einladungsprozesses Geschwindigkeit & Feedback

Der Einladungsprozess war zu langsam und gab zu wenig Rückmeldungen, daher waren die Ziele für dieses Arbeitspaket eindeutig:

1. Sofortige Rückmeldungen bei Interaktionen
2. Mehr Rückmeldungen über Systemoperationen und deren Laufzeit
3. Langlaufende Operationen sind abbrechbar
4. Langlaufende Operationen sollen beschleunigt werden

Die Auswirkungen von Fortschrittsanzeigen, auf die Zufriedenheit der Nutzer, wurde unter anderem von Myers (1985) untersucht, wobei sich ergab, dass Nutzer Fortschrittsanzeigen für langlaufende Prozess stark bevorzugen und insgesamt als nützlich ansehen. Frederickson et Al. (1993) beschreiben weiter, dass die tatsächliche Dauer des Prozesses eher weniger Auswirkungen auf die Zufriedenheit der Anwender hat. Menschen würden sich nicht auf konsistente, lineare Art und Weise erinnern, aber vielmehr Ereignisse selektiv wahrnehmen. In diesem Zusammenhang beschriebene Effekte, wie „Duration Neglect" (Vernachlässigen der tatsächlichen Dauer eines Prozesses) und „peak-and-end effects" (Fokus auf das Erlebnis des Prozessendes) finden sich nicht nur im Zusammenhang mit der Mensch-Computer-Interaktion sondern in vielen Bereichen, darunter Medizin, Wirtschaft und Werbung (Vgl. [Frederickson1993], [Myers1985], [Harrison2007]).

Harrison et Al. untersuchen in „Rethinking the Progress Bar" mögliche Verbesserungen der User Experience durch Manipulation der Darstellung von Fortschrittsbalken für Prozesse deren Ende nicht genau vorhergesagt werden kann, zum Beispiel durch dynamische Endbedingungen (ein Beispiel wäre die Defragmentierung von Festplatten). Nutzer würden negatives Verhalten der Fortschrittsanzeige, z.B. in Form von Stagnation (der Fortschrittsbalken beweg sich nicht mehr), oder inkonsistentem Verlauf der Fortschrittsanzeige zu Beginn des Prozesses eher tolerieren, als gegen Ende. Für linear laufende Prozesse, mit stabilen Endbedingungen seien herkömmliche Fortschrittsbalken zu empfehlen. Für Prozesse mit stark variablem Abschnitten sollte geprüft werden, ob die Reihenfolge der Prozessabschnitte geändert werden kann, sodass die variablen Abschnitte zu Beginn des Prozesses bearbeitet werden, um in danach folgenden Abschnitten ein leicht vorherzusagendes Verhalten der Fortschrittsanzeige zu bieten (Vgl. [Harrison2007]).

Bei der systematischen Integration von mehr Fortschrittsanzeigen in Saros wurden diese Empfehlungen berücksichtigt. Es war hierbei aber zeitlich nicht mehr möglich, dem Ansatz zu folgen, tatsächlich erfolgten Fortschritt zwischen zu speichern, und die Fortschrittsanzeige gegen Ende beschleunigt darzustellen. Derartige Manipulationen müssten entweder direkt in den Util-Klassen erfolgen, die beispielsweise eine ZIP-Datei entpacken, oder die Checksummen berechnen,

oder aber über eine eigene Kapselung der Job-Klassen von Eclipse (siehe Anhang 7.3 vorgenommen werden.

Bei der Bearbeitung dieses Arbeitspakets wurden die Ziele in Bezug auf die verletzten Heuristiken schrittweise bearbeitet, die im Folgenden beschrieben werden.

Ziel 1: Sofortiges Feedback bei Interaktionen (N1 Sichtbarkeit des Systemstatus)

Durch asynchrones Übernehmen der Selektion aus dem offenen Navigator-artigen View der aktuellen Perspektive in den Wizard, zum Starten einer Sitzung bzw. zum Hinzufügen eines Projekts zu einer Sitzung, erscheint dieser Wizard nun sofort, ist aber noch blockiert, bis die Selektion angewandt ist. Dadurch erhält der Anwender sofort Rückmeldung, dass die gewünschte Aktion ausgelöst wurde. Mangelndes Feedback über den Erfolg von Interaktionen mit Saros war zuvor einer der größten Mängel im Einladungsprozess.

In diesem Zusammenhang wurde auch das Problem des flackernden Mauszeigers behoben, das beim Öffnen des Wizards über die Kontextmenüs auftrat. Durch Einsatz der von Eclipse bereitgestellten *BusyIndicator*-Klasse, wird der Mauszeiger des Nutzers auf einen Zeiger, der Systemaktivität anzeigt, festgesetzt (bei Windows z.B. die Sanduhr, bzw. der sich drehende Kreis).

Als nächsten Schritt wurden die Systemaktivitäten schrittweise mit Fortschrittsanzeigen ausgestattet, sofern die Möglichkeit bestand, dass sie eine Laufzeit von mehr als 2 Sekunden haben können. In diesem Fall wurden Fortschrittsbalken angezeigt bzw. zunächst die Aktivitäten in die Eclipse-Jobs-Klassen gekapselt, um dies zu ermöglichen. Dadurch erhalten Nutzer mehr, und aussagekräftige Rückmeldungen über die Aktivitäten des Systems und deren Laufzeit. Eine Übersicht der Möglichkeiten die hierfür von Eclipse bereitgestellt werden, finden sich im Anhang 7.3.

Ziel 2: Mehr Feedback über Systemoperationen und deren Laufzeit

(N1 Sichtbarkeit des Systemstatus)

Für den Sitzungsaufbau muss zunächst der Workspace in Eclipse aktualisiert werden, und es wird eine Liste aller Dateien aus den gewählten Projekten generiert. Dies kann etliche Sekunden dauern, hat aber eine vorher nicht zu bestimmende Laufzeit. Daher wird bei der Kapselung in einen Eclipse-Job der Parameter *IProgressMonitor.UNKNOWN* verwendet, sodass ein Fortschrittsbalken dargestellt wird, der zum Beispiel bei Windows, zwischen Links und Rechts im Dialog, hin und wandert, am anzuzeigen das ein Prozess aktiv ist, aber nicht bestimmt werden kann, wie weit der Prozess fortgeschritten ist.

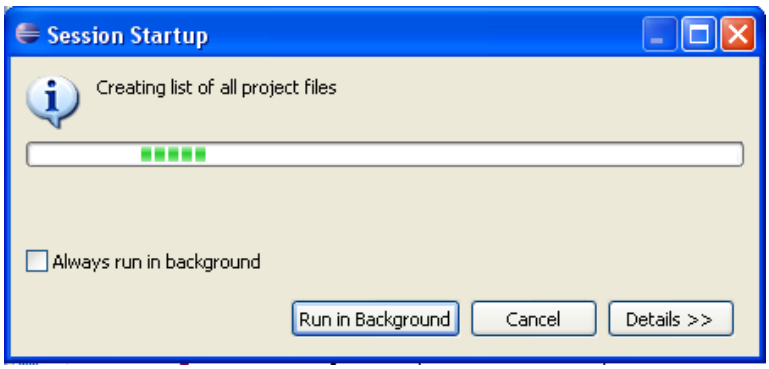

Abbildung 26: Darstellung eines Fortschrittsbalkens für einen Prozess mit unbekannter Dauer

In diesem Prozess gab das System zuvor keinerlei Rückmeldung. Erst durch das Kapseln als Eclipse-Job, ist es dem Nutzer somit ermöglicht worden, diesen langlaufenden Prozess abzubrechen und die Kontrolle zu behalten (N3).

Bei der später folgenden Berechnung der Checksummen wird zunächst eine Liste der zu verarbeitenden Dateien ermittelt, damit die Laufzeit der Operation genauer bestimmt werden kann (diese Änderung erfolgte durch einen sehr technisch motivierten Teamkollegen). Der Fortschrittsbalken bewegt sich nun durchgehend und zeigt tatsächlich den Fortschritt an (keine fest codierten Prozentzahlen mehr). Zudem wird angezeigt, dass gerade eine Checksumme berechnet wird, und für welche Datei. Nutzer wünschten sich mehr Informationen über die tatsächliche technische Aktivität von Saros. Da die Zielgruppe Software-Entwickler sind, wird es als sinnvoll angesehen, technische Fachbegriffe zu verwenden, da diese dem Nutzer mit großer Wahrscheinlichkeit geläufig sind.

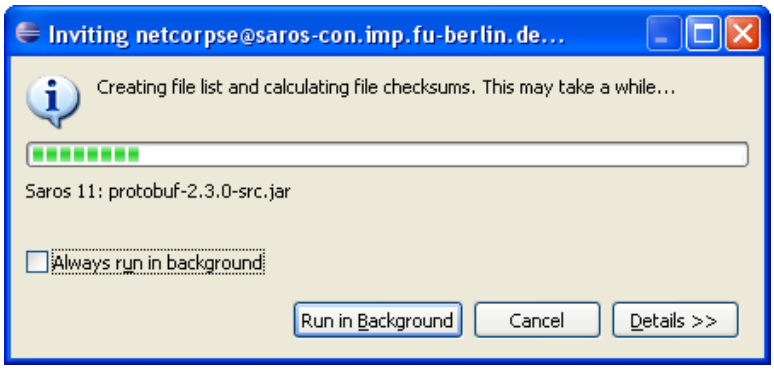

Abbildung 27: Linearer Verlauf des Fortschrittsbalkens für die Checksummen-Berechnung

In den darauffolgenden einzelnen technischen Abschnitten des Einladungsprozess wurden „interne“ Meldungen aus den Fortschrittsanzeigen entfernt und durch untechnische Rückmeldungen ersetzt (N2. Übereinstimmung zwischen System und realer Welt). So wurde beispielsweise der unverständliche Text „Waiting for user list request“ ersetzt durch „Waiting for <username> to choose project location“, wodurch verdeutlicht wird, dass der Prozess jetzt stagniert, bis der eingeladene Nutzer gewählt hat, wie und wo er die eingehenden Projektdateien ablegen möchte, und das System nicht aufgrund eines Fehlers hängen geblieben ist.

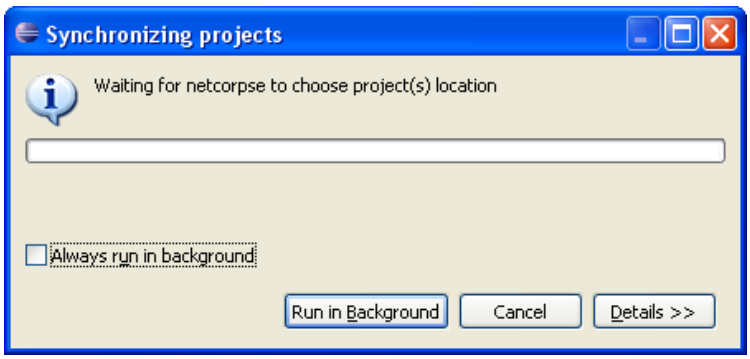

Abbildung 28: Wartezustand beim Host

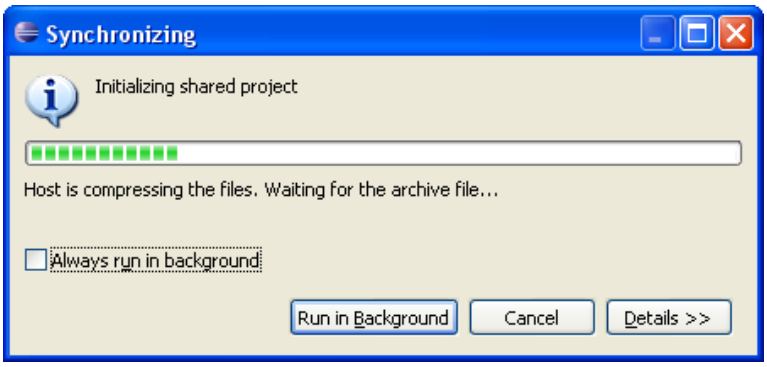

Abbildung 29: Wartezustand beim Empfänger

Im weiteren Verlauf des Einladungsprozesses wird nun dem Nutzer detailliert angezeigt und fortlaufend errechnet, wie hoch die Übertragungsgeschwindigkeit beim Senden der ZIP-Datei mit den Projektdateien ist.

Auch enthalten sind Größenangaben der Gesamtdateigröße, und einer Angabe, wie viele Daten bisher übertragen wurden, sowie eine Zeitangabe, die die voraussichtlich verbleibende Laufzeit des Übertragungs-Prozesses anzeigt. Diese Anzeige wird laufend aktualisiert, basierend auf der durchschnittlichen Übertragungsdauer für die bisherige Datenmenge. Somit ist sichergestellt, dass auch bei starken Veränderungen der Übertragungsgeschwindigkeit die voraussichtliche Dauer korrigiert wird, ohne sich hierbei, bei gelegentlichen Geschwindigkeitsschüben, zu schnell zu verändern.

Durch die neue Anzeige der zu erwartenden Restdauer bekommen die Saros-Anwender die Möglichkeit, sich anderweitig zu beschäftigen, bis der Einladungsprozess abgeschlossen ist. Zuvor war den Nutzern unklar wie lange der Prozess noch dauern kann, weshalb häufig die Fortschrittsbalken beobachtet wurden, bis der Prozess abgeschlossen war.

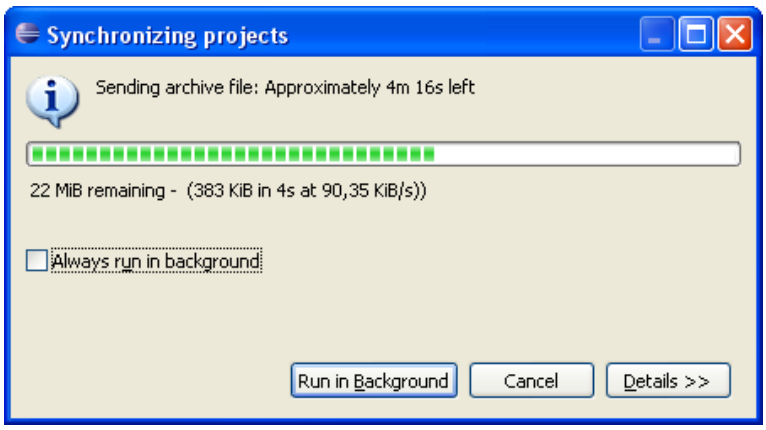

Abbildung 30: Detaillierte Fortschrittsanzeige beim Host

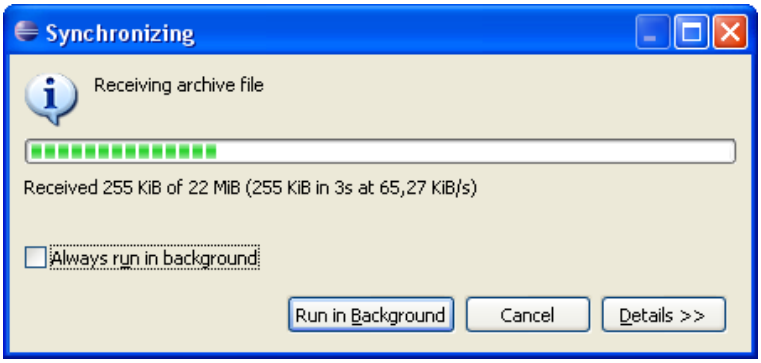

Abbildung 31: Detaillierte Fortschrittsanzeige beim Empfänger

Bei der Berechnung der Geschwindigkeit und der Dateigrößen wurden zunächst die vorhandenen Methoden aus der *Util*-Klasse von Saros verwendet. Hier war zunächst nicht aufgefallen, dass dort die technischen, weniger bekannten, IEC-Präfixe[24] und Berechnungsmethoden für die Byte-Angaben verwendet wurden, die bei dem Akzeptanz-Test beim März-Release als ungewöhnlich bemängelt wurden: 1 Kibibyte (KiB) = 1024 Byte, 1 Mebibyte (MiB) = 1024 · 1024 Byte [25]

Die Verwendung dieser „ungewöhnlichen" (aber aus Informatiker-Sicht korrekten) Einheiten verstößt gegen die Heuristik N2:

> N2. Übereinstimmung zwischen System und realer Welt
> Das System sollte die Sprache des Benutzers sprechen. Dafür soll es keine systemorientierten Begriffe, sondern Wörter und Formulierungen, die dem Nutzer geläufig sind, nutzen. [Nielsen1994]

Alle Berechnungen innerhalb von Saros wurden daher umgestellt auf die gängigen SI-Einheiten zur Basis 10, womit sich die folgenden Werte ergeben:

1 Kilobyte (kB) = 1000 Byte, 1 Megabyte (MB) = 1000 Kilobyte = 1000 · 1000 Byte.

---

[24] IEC: International Electrotechnical Commission, Normungsorganisation für Elektrotechnik und Elektronik

[25] Anschauliche Erläuterungen zu den Präfixen für die Präfixe finden sich auf http://de.wikipedia.org/wiki/Byte

Nach diesen Anpassungen ging es daran, dem Nutzer die Gelegenheit zu geben, zu verstehen, welche Prozesse in Saros eine lange Laufzeit haben. Durch die Anzeige, welche Prozesse es gibt und der Möglichkeit zu erkennen, welche Aktivitäten eine lange Laufzeit haben, wird dem Nutzer die Möglichkeit gegeben, das System besser seinen Bedürfnissen anzupassen. So kann der Nutzer erkennen, dass z.B. die Ermittlung der SVN Revisionsnummern zeitintensiv ist, und somit für eine höhere Performance deaktiviert werden könnte, wenn kein Mehrwert durch die Verwendung der SVN Integration vorliegt (z.B. weil die Verbindung zum SVN Repository langsamer ist als eine direkte Datenübertragung).

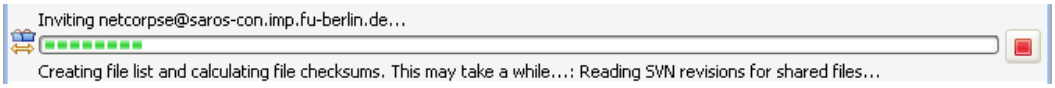

Abbildung 32: Ein Hinweis darauf, dass die Ermittlung der SVN Revisionen Zeit kostet

In Bezug auf die SVN-Integration wurde zudem durch das Implementieren eines *RemoteProgressMonitor* erreicht, dass auch der Host bei einem SVN-Checkout-Prozess (der häufig mehrere Minuten dauert) eigenständig ableiten kann, wie lange der Prozess noch dauern könnte.

Der *RemoteProgressmonitor* ist in der Lage, einen beliebigen lokalen *ProgressMonitor* von Eclipse (z.B. in einem Job) zu anderen Saros-Nutzern zu spiegeln. Dies wurde eingesetzt, um dem Host (dem einladenden Nutzer) den Fortschrittsbalken für SVN Checkout-Prozesse anzuzeigen. Vor dieser Anpassung war für den Host nichts zu sehen, wenn der eingeladene Nutzer sich entschieden hatte, das ankommende Projekt per SVN über Saros abrufen zu lassen. Dadurch, dass dieser Fortschrittsbalken nun auch beim Host erscheint kann dieser Rückschlüsse ziehen über die zu erwartende Restdauer des Prozesses. So haben beispielsweise Saros-Entwickler ein gutes Gefühl dafür, wie lange ein SVN Checkout noch dauern wird, wenn dieser die LIB-Dateien (z.B. JAR-Bibliotheken) erreicht hat.

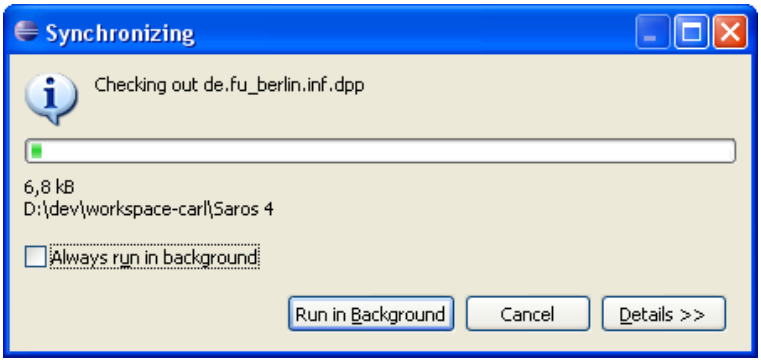

Abbildung 33: Gespiegelte Fortschrittsanzeige für Host und eingeladenen Nutzer bei einem SVN Checkout über Saros

Durch die Änderungen in Hinsicht auf die Fortschrittsanzeigen wurde gleichzeitig das Ziel 3 abgearbeitet, das sich mit der Heuristik N3. Benutzerkontrolle und Freiheit beschäftigt.

Ziel 3: Langlaufende Operationen sind abbrechbar

Die Kapselung der Prozesse über die Job-API von Eclipse ermöglicht es, den Prozess zeitnah abzubrechen, da die Dialoge eine Abbrechen Schaltfläche haben. Durch Hinzufügen einer sogenannte „Job-Family“ sind die Prozesse für Eclipse nun zudem gruppierbar (werden zusammengefasst wenn man in einem Fortschrittsdialog auf „Details“ klickt) und die Eclipse-Funktionalität „Always run in background“ funktioniert nun (N4. Konsistenz und Standards)

Da nun fast alle Operationen von Saros deutlich über den Fortschritt, und wenn möglich die verbleibende Dauer, informieren, ging es nun an das Erreichen des nächsten Ziels, der Geschwindigkeits-Optimierung von Saros.

Ziel 4: Langlaufende Operationen sollen beschleunigt werden (N7 Flexibilität und effiziente Nutzung)

Saros beinhaltete mehrere Ressourcenverschwendungen, die dazu führten, dass Prozesse unnötig lange dauern. Verbesserungen der Performance von Saros wurden unter anderem durch das Entfernen mehrfach redundanter, überflüssiger Methodenaufrufe innerhalb des Einladungsprozesses erreicht.

Die Laufzeit für die Übernahme einer existierenden Selektion aus der aktuellen Perspektive sowie nachträglicher Änderungen der Selektionen bei der Ressourcen-Auswahl wurde deutlich verkürzt. So sank die durchschnittliche Wartezeit bei der Auswahl von Projekten mit vielen Dateien (z.B. Saros mit 3000 Dateien) im Einladungs-Wizard auf weniger als 3 Sekunden, während diese zuvor bei 10 Sekunden oder mehr lag (auf einem 4 Monate alten 4-Prozessor-Kern „Samsung X9“ Laptop mit Solid State Disk).

Zusätzlich wurde durch Verwenden der von Eclipse bereit gestellten *BusyIndicator*-Klasse, der Mauszeiger auf den Mauszeiger-Typ für blockierende Operationen gesetzt. Dies signalisiert dem Nutzer, dass das System gerade aktiv ist und verhindert, dass der Nutzer (wie bei dem vorherigen Mauszeiger für Hintergrundaktivität) bereits anfängt im Wizard zu klicken, obwohl dieser tatsächlich noch blockiert ist für Nutzerinteraktionen.

Weitere Optimierungen konnten durchgeführt werden, nachdem identifiziert wurde, dass Saros bei Einladungen von mehreren Nutzern gleichzeitig redundant Checksummen für alle Dateien berechnet, und dies für jeden Nutzer in einem parallel laufenden, separaten Thread. Hierzu wurde ein entsprechender Bug im Bugtracker angelegt, und das Problem wurde durch einen Teamkollegen durch die Integration eines Zwischenspeichers für berechnete Checksummen von Dateien behoben. Somit verbesserte sich die Geschwindigkeit der Checksummenberechnung für den Einladungsprozess bei N eingeladenen Nutzern (N>2) um mehr als das N-fache. Da die redundanten Berechnungen zudem parallel abliefen ist der Performancegewinn immens, da die parallelen Schreibzugriffe auf die Festplatte so deutlich vermindert werden, und die Gesamtanzahl der Festplatten-Zugriffe deutlich sinkt.

Nach der Bearbeitung des Arbeitspakets zur Optimierung der System-Rückmeldungen wurden die Wizards des Einladungsprozesses optimiert.

5.1.2 Arbeitspaket: Einladungsprozess Wizards 1

Im alten Wizard für die Projekteinladung waren mehrfach redundante, und verwirrende Beschriftungen und Anleitungstexte beinhaltet, die den Nutzern in Tests nicht verdeutlichen konnten, dass auch das Teilen einzelner Dateien möglich ist. Da dies eine zentrale Funktion von Saros ist, wurde dieser Wizard mit hoher Priorität überarbeitet.

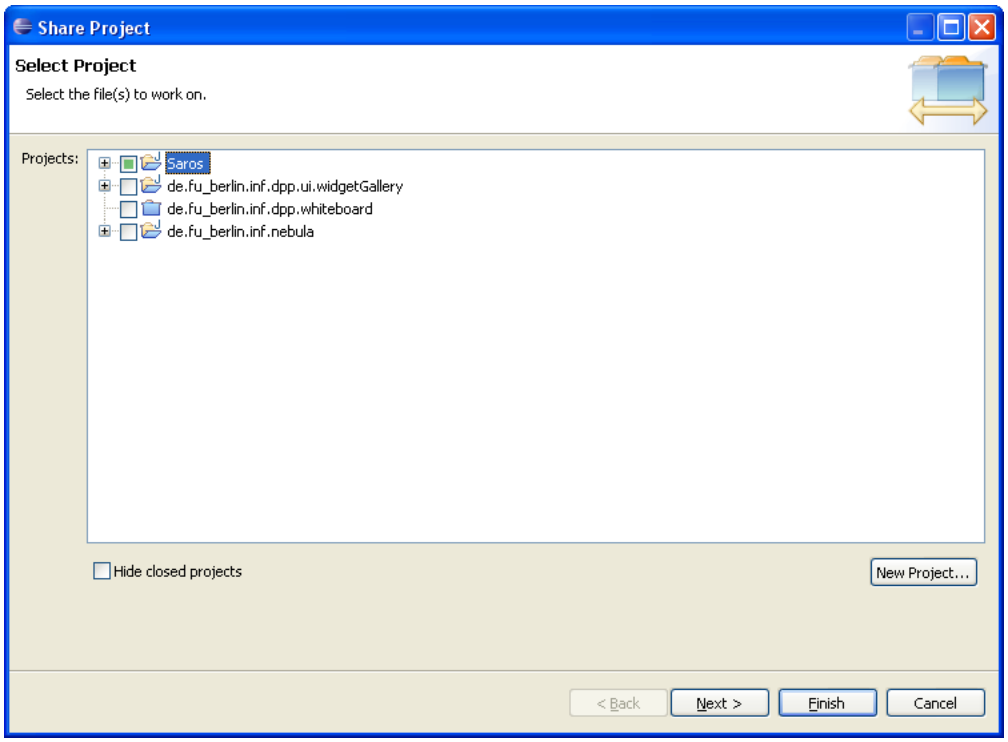

Abbildung 34: Der alte Wizard für die Projektselektion

Für die Veränderungen an den Wizards im Einladungsprozess wurden zunächst Ziele definiert:

Ziel 1: Erlernbarkeit des Systems verbessern

- Es soll deutlich werden, dass das Selektieren einzelner Dateien für einen Sitzungsaufbau genügt
- Es soll klar sein, das es ein Rechtekonzept in Saros gibt und Nutzer zunächst gleichberechtigt mit Schreibzugriff starten (ohne, dass hierfür eine Hilfe gelesen werden muss)

Ziel 2: Mehr Kontrolle über den Start der Sitzung durch den Host

- Der Host soll bereits bei der Einladung festlegen können, wer Schreibrechte bekommt
- Der Host kann andere Nutzer im Follow-Modus starten lassen für ein schnelleres Starten

Ziel 3: Der Einladungsprozess soll schneller und bequemer sein

- Nutzer können die letzte verwendete Ressourcen-Selektion einfach wieder herstellen
- Nutzer können komplexe Selektionen (Partial Sharing), unter selbst festgelegten Namen abspeichern und diese bei Bedarf wiederherstellen

Ziel 4: Vermeidung von Frustration durch Fehlbedienung

- Versehentliche getätigte Selektionen sollen rückgängig gemacht werden können (Heuristik N3 Benutzerkontrolle und Freiheit)

Hieraus entstand in mehreren Iterationen ein neues Layout des Wizards. Es wurden Funktionalitäten in den Wizard integriert, die neue Nutzer mit großer Wahrscheinlichkeit wahrnehmen werden, und somit durch die Darstellung der Funktionen einen Hinweis bekommen, dass eine derartige Funktionalität existiert, was wiederum das explorative Erkunden der Funktionalität herbeiführen könnte (Vgl. [Dix2003]).

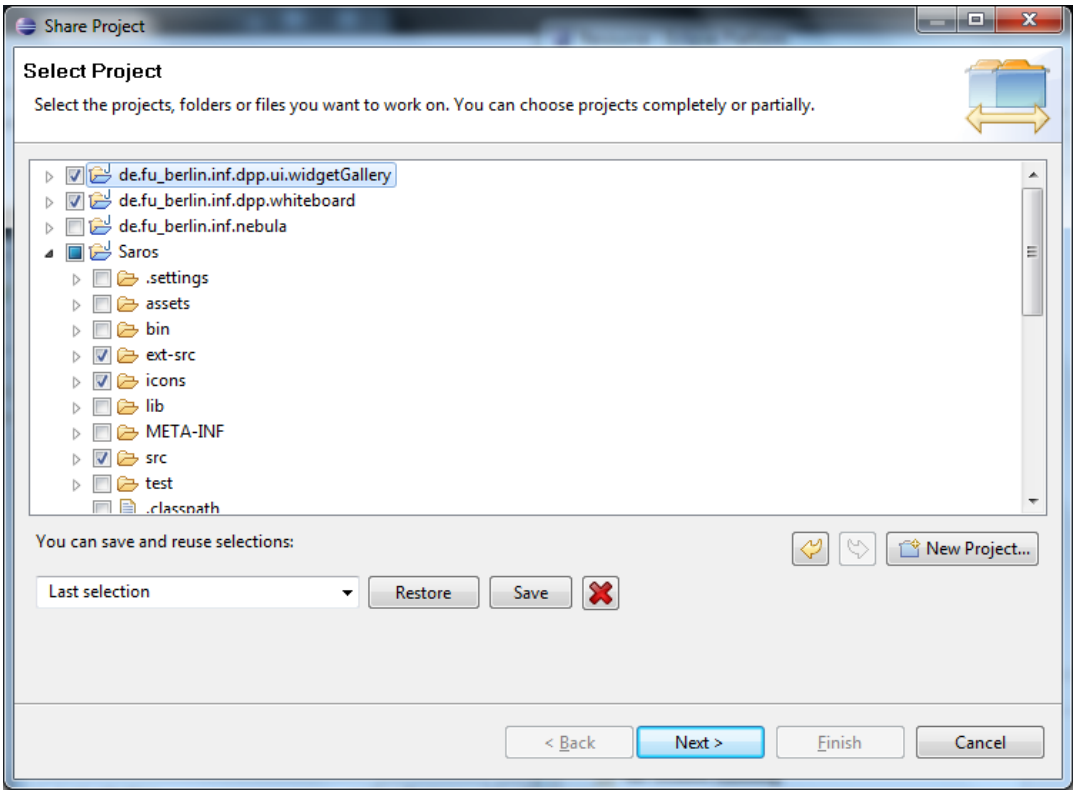

Abbildung 35: Der neue Wizard zum Auswählen der Ressourcen für das Projekt

Im Einzelnen wurden folgende Anpassungen vorgenommen:

N7. Flexibilität und effiziente Nutzung
- Selektionen können gespeichert werden
- Selektionen können wiederhergestellt und gelöscht werden
- Selektionen können überschrieben werden

N3. Benutzerkontrolle und Freiheit

- Rückgängig / Wiederholen (Undo/Redo) wurde in die Projektselektion integriert, unter Verwendung der Eclipse Standard-Icons für diese Aktionen.
- Die Schaltflächen werden nur aktiviert, wenn die hinterlegten Aktionen auslösbar sind
- Tooltips für die Schaltflächen weisen auf deren Funktionalität hin

N4. Konsistenz und Standards

- Die Schaltfläche „New Project" erhält das Eclipse-Standard Icon für diese Funktionalität (Angleichung an Eclipse Look & Feel)
- Schaltflächen, die Standard-Eclipse-Aktionen auslösen, sollten mit dem entsprechenden Icon ausgestattet werden
- Die Sortierung in der Baumdarstellung entspricht der Sortierung die von Eclipse verwendet wird

N8. Ästhetisches und minimalistisches Design

- Redundante Beschriftungen wurden entfernt. So wurde das Label „Projects:" entfernt, da es Platz verbraucht, ohne einen Mehrwert zu bieten. Vielmehr war bei der Beobachtung der Probanden zu vermuten, dass die Nutzer die anderen Beschriftungen (z.B. in der Kopfzeile des Wizards) nicht lesen, da sie direkt im Inhaltsbereich (der grauen Fläche) des Wizards anfangen zu lesen.

Die angestrebte Anordnung der Elemente, so dass die Undo/Redo/New Project nach rechts oben, oberhalb der Baumdarstellung platziert sind, wurde nicht mehr erreicht, da hierzu die Klassenstruktur der Viewer-Klassen komplett geändert werden müsste. Weitergehende Verbesserungen des Layouts stehen aus Konsistenz-Gründen (zum Angleichen an die gängige Eclipse-Darstellung) noch aus.

Denkbar wäre hier auch der Empfehlung zur „Diagonal Balance" zu folgen, die in [Tidwell2011, S.176] empfohlen wird, bei der Elemente in einer diagonalen Anordnung so angelegt werden, dass für den Betrachter wieder ein Gleichgewicht entsteht. Hierzu könnten die Projektselektion nach links oben verschoben, und die Undo/Redo Funktionalität könnte rechts oben angeordnet werden. Hierdurch würde die Aufteilung auf einer virtuelle Diagonalen auf den Nutzer ausbalanciert wirken. Links oben die gespeicherten Selektionen, rechts unten die Schaltflächen zum Fortfahren. Allerdings wäre hier zu prüfen, ob eine Beschriftung der Baumdarstellung wieder notwendig wird, da nun die räumliche Nähe zur Beschreibung im Wizard-Kopf nicht mehr ausreichend gegeben wäre (Vgl. „Proximity", [Tidwell2011, S.139])

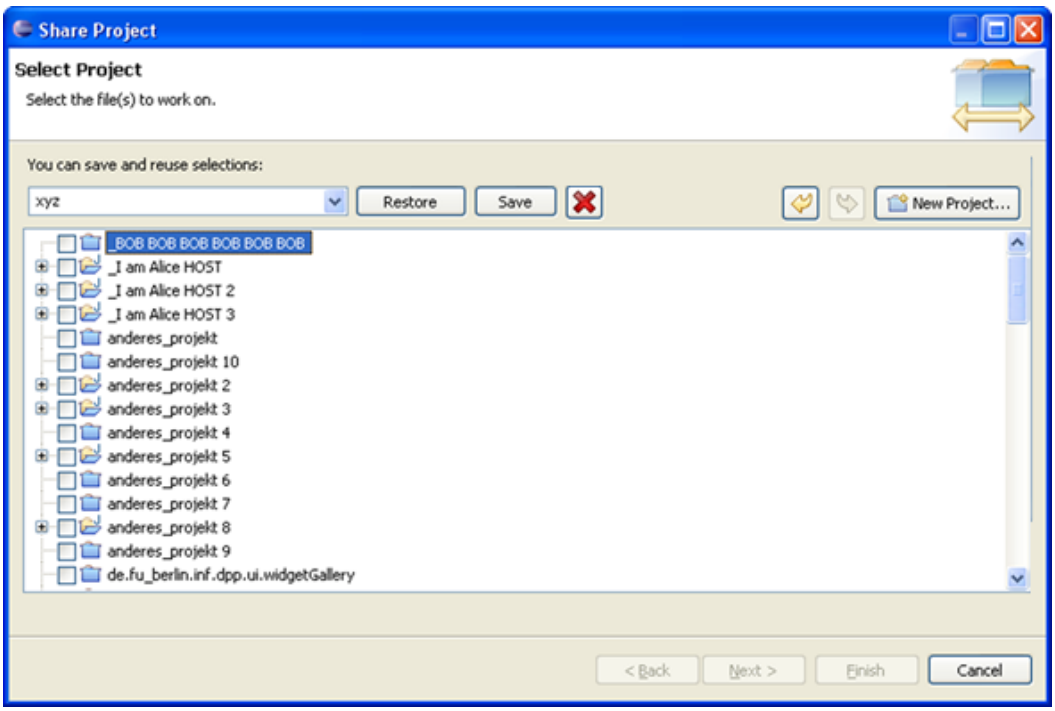

Abbildung 36: Layout-Entwurf für Projekt-Selektion im Einladungs-Wizard

Nach dem der Wizard für die Projektselektion überarbeitet wurde, ging es an die Auswahl der Buddies als Teilnehmer einer Sitzung. Dieser wurde gemäß der Heuristik N5 (Fehler vermeiden) so verändert, dass nur noch Buddies eingeblendet werden, die online sind. Da Änderungen dieser Einstellung automatisch gespeichert werden, hat der Nutzer die Möglichkeit auch wieder den alten Zustand herzustellen, da es gelegentlich Probleme gibt mit der Erkennung des „Online"-Status eines Buddies.

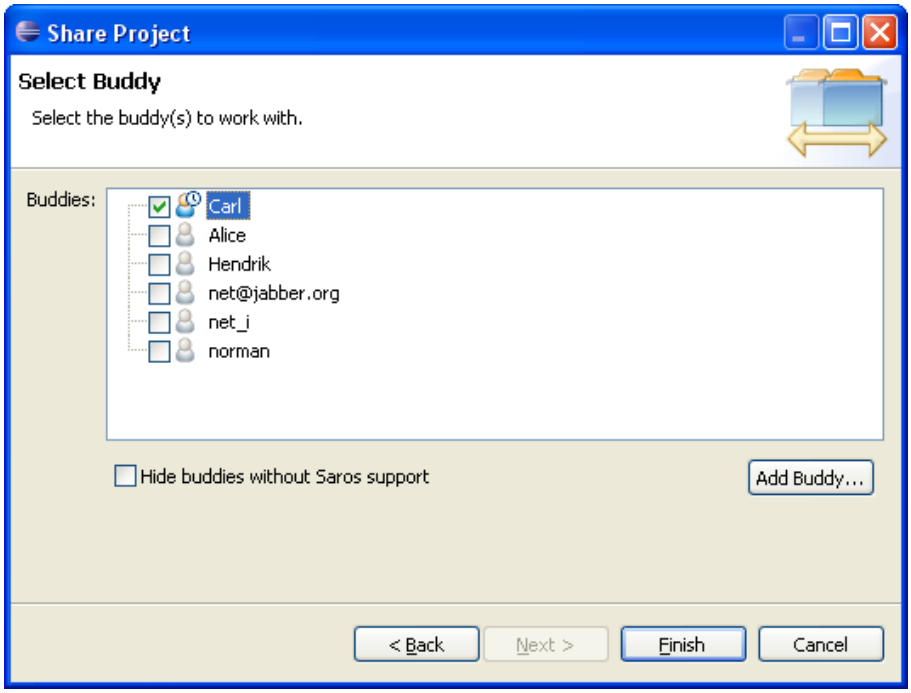

Abbildung 37: Alte Auswahl der Buddies mit allen Nutzern

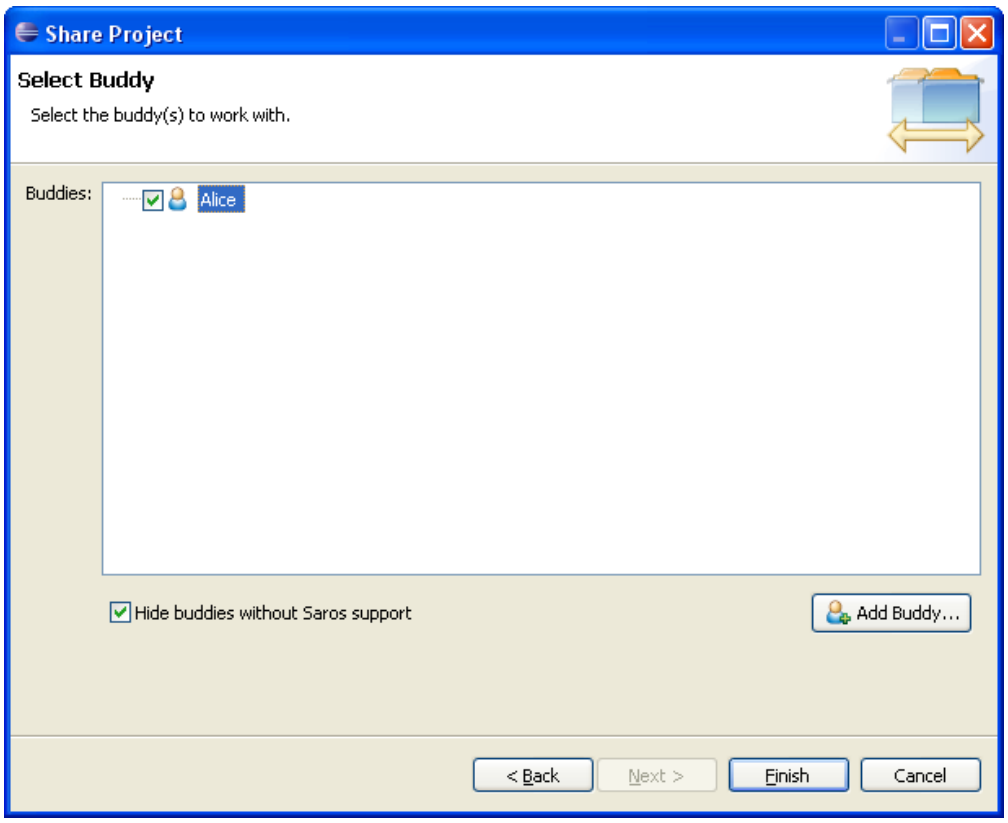

Abbildung 38: Die überarbeitete Auswahl der Buddies

Die Wizard Seite ist somit wesentlich übersichtlicher, da in den Standard-Einstellungen nur noch Buddies dargestellt werden, die mit Saros online sind und somit auch eingeladen werden können.

Im Rahmen der Anpassung für N4 Konsistenz und Standards wurde auch der Schaltfläche „Add Buddy“ das passende Icon hinzugefügt. Weitere Anpassungen dieses Wizards erfolgten im Rahmen des nächsten Arbeitspakets.

5.1.3 Arbeitspaket: Einladungsprozess Wizards 2

Um dem einladenden Nutzer vor Absenden der Einladung in einer Zusammenfassung (wie auch bei der Einrichtung von Saros) anzuzeigen, welche Aktion gleich ausgeführt wird, wurde in einem Prototypen eine zusätzliche Seite in den Einladungs-Wizard integriert.

Hier sind folgende Informationen aufgeführt:

- Welche Nutzer werden eingeladen
- Welche Projekte sind Bestandteil der Sitzung

Zudem kann der Host hier wählen, ob ein Nutzer nur mit Leserechten, ohne Schreibzugriff, in die Sitzung eingeladen werden soll, und ob der Nutzer automatisch dem Host folgen soll, nach Akzeptieren der Einladung. Auch gibt es einen Hinweistext, der beschreibt, was der Nutzer in dieser Seite tun kann.

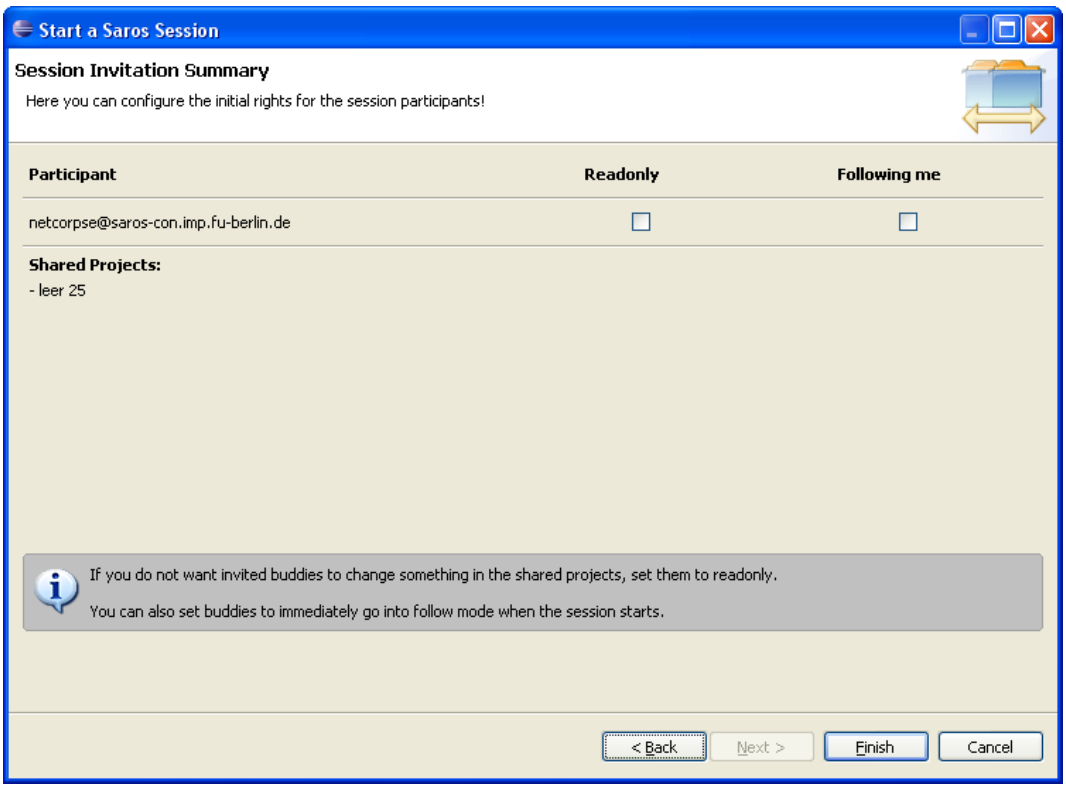

Abbildung 39: Zusammenfassung der Einladung als letzte Seite im Wizard

Bei Einladung von Buddies in eine Session war es zuvor dem Host nicht möglich, im Einladungsprozess zu definieren, welche Rechte die eingeladenen Benutzer haben sollen. So hat jeder Benutzer initial Schreibrechte auf alle Dateien die innerhalb der Saros Session geteilt werden, auch wenn der Host dies nicht möchte.

In Bezug auf das fehlende Changelog ergibt sich hierbei das Problem, dass ein Sitzungsteilnehmer Dateien modifizieren kann, bevor der Host der Session die Möglichkeit hat, allen Teilnehmern die Schreibrechte zu entziehen, und die Änderungen automatisch auch im Workspace des Hosts persistiert werden.

Anmerkungen zur Relevanz dieses Problems:

Man könnte argumentieren, dass es doch sehr unwahrscheinlich sei, dass ein Sitzungsteilnehmer, ohne dies explizit zu kommunizieren, eine Datei modifiziert, bevor ihm die Schreibrechte durch den Host entzogen werden. Hierbei sollte man allerdings bedenken, dass auch Modifikationen, die nicht durch explizite Benutzerinteraktion ausgelöst wurden, synchronisiert werden. Direkt nach der erfolgreichen Übertragung der Projektdateien kann ein automatischer Build eines übertragenen Projekts von Eclipse ausgelöst werden. Als Beispiel sei hier ein Build-Script (z.B. ein ANT-Script) gewählt, dass eine JAR-Datei erstellt, die nach der Erstellung durch Saros automatisch zu allen anderen Sitzungsteilnehmern synchronisiert wird, was insbesondere bei Partial Sharing zu Problemen führen kann. (Zusätzlich angenommen der eingeladene Sitzungsteilnehmer nutzt Java 1.6 womit die Jar-Datei bei anderen Sitzungsteilnehmern mit Java 1.4 oder älter nicht kompatibel wäre, könnte dies zu handfesten Problemen führen).

Auch ungewollte Modifikationen, beispielsweise das versehentliche Löschen einer Datei, oder viel wahrscheinlicher: versehentliches Drag & Drop im Package Explorer, führen zu Modifikationen im Workspace, die Saros gewissenhaft zu allen Sitzungsteilnehmern übertragen wird.

Besonders hervorzuheben ist bei diesem Problem, dass es eine Diskrepanz zwischen dem mentalen Modell des Benutzers, in Erwartung von „Session-Host hat die Kontrolle über die Sitzung“ und der tatsächlichen Systemstruktur gab, die hierdurch aufgelöst werden sollte.

Dieser erste Entwurf wurde allerdings aus mehreren Gründen wieder verworfen:

- Nutzer, die Saros kennen, erreichen diese Wizard-Seite nicht
    - Da die Optionen dort alle optional sind, ist es nicht nötig im Wizard auf „Weiter >“ zu klicken – der Nutzer kann nach der Wahl der Buddies wie gewohnt auf „Finish“ klicken. Die Sichtbarkeit dieser Wizard Seite ist somit sehr gering und erfüllt damit ihren Zweck nicht.
- Bei der Nutzung wurden entweder alle oder keine der Checkboxen einer Spalte angehakt.
    - Es ist schwer, einen realistischen Anwendungsfall zu konstruieren, in dem man einen Nutzer als Readonly einlädt, und einen anderen Nutzer mit Schreibrechten.
- Die Seite aktualisiert sich nicht automatisch. Wenn der Buddy zwischenzeitlich offline geht, dann kann die Einladung abgeschlossen werden, führt aber zu einem Fehler. Dies ist inakzeptabel.

Es würde wenig Sinn machen, Optionen auf dieser Seite in Radio-Buttons ohne Vorauswahl zu verwandeln, sodass die Seite nicht mehr optional ist, da dies die Einladung verlangsamen würde. wurde hier komplett neu konzipiert. Daraufhin entstand die Idee einer Integration einer „initialen Sitzungskonfiguration“ in die Seite, auf der die Buddies eingeladen werden.

Der Host kann hier für alle eingeladenen Nutzer wählen, ob dies automatisch in den Followmodus wechseln oder ob alle Nutzer nur Lesezugriff erhalten sollen (der ja nachträglich geändert werden kann über das Kontextmenü).

Ein Problem stellte sich bei internen Tests dieses Prototypen heraus: Eingeladene Nutzer starteten automatisch im Follow-Modus, allerdings bemerkten sie dies nicht, da sie nicht darüber informiert wurden (hier hätte vermutlich eine Ballon Benachrichtigung geholfen, oder ein Dialog, der den Nutzer fragt, ob dieser tatsächlich im Follow-Modus starten möchte).

Leider war es mehr nicht möglich, dies mit anderen Saros-Nutzern zu evaluieren, und es blieb es bei einem Prototyp für diese Funktionalität. Da unter keinen Umständen ungetestete Änderungen im finalen Saros landen sollten, da dies in der Vergangenheit genug Probleme bereitet hatte, verbleiben diese Änderungen im Review-Board, bis erfolgreich evaluiert wurde, wie diese Funktion von den Nutzern aufgenommen wird.

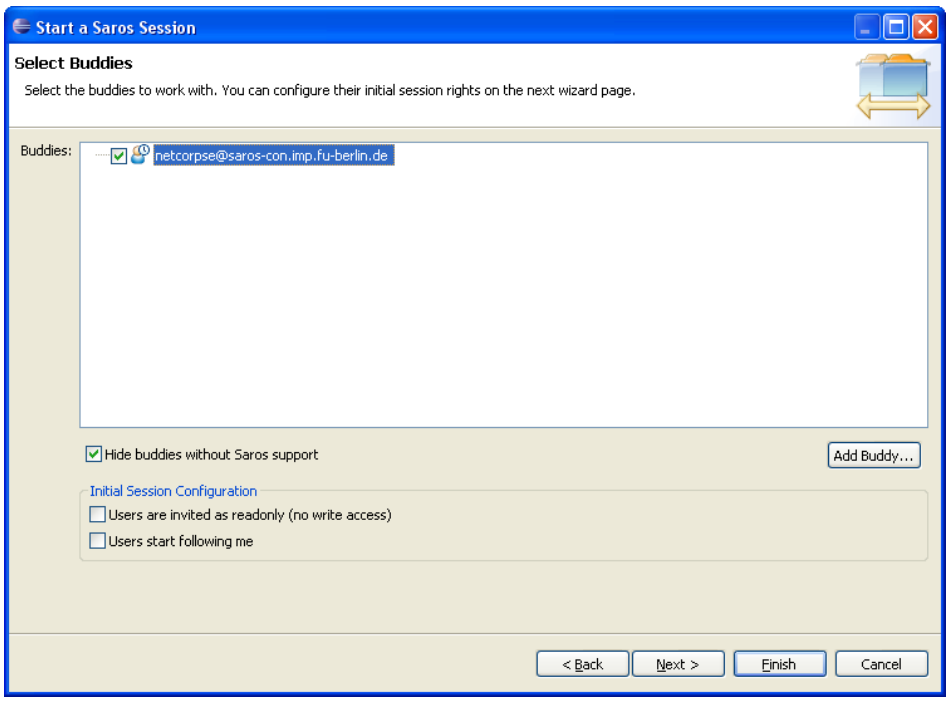

Abbildung 40: Die überarbeitete Auswahl der Sitzungsteilnehmer

5.1.4 Arbeitspaket: Einladungsprozess Wizards 3

Ein weiterer problematischer Wizard war der Dialog für eingehende Projekte, nach dem Annehmen einer Einladung. In diesem Dialog wird ein TabControl angezeigt, mit einem Tab pro Projekt das Bestandteil der Einladung ist. In Tidwell (2011) wird beschrieben, dass Tabs gut dafür geeignet sind, um viele heterogene Inhalte in einer einzelnen Ansicht zu zeigen. Auch seien Tabs gut geeignet für sehr ähnliche Inhalte, die in Module unterteilt sind und dabei möglichst gleiche Abmaße (Höhe / Breite) haben. Die Anzahl der Module sollte hierbei auf weniger als 10, besser weniger, beschränkt sein [Tidwell2011]. Das Problem hierbei ist jedoch: Die Menge an Modulen die in Tabs dargestellt werden, sollte möglichst statisch sein. Neue Tab-Seiten sollten nicht regelmäßig hinzukommen oder wegfallen, was hier nicht gegeben ist. Je nachdem, welche Projekte ein Nutzer für die Sitzung verwendet, erscheinen hier gänzlich andere Tabs.

Hierzu ist noch eine Anmerkung notwendig: In den Usability-Tests wurde fast kein Kontakt mit der Funktionalität für das Wiederverwenden existierender Projekte beobachtet. Nutzer hatten zwar in einigen Sitzungen bereits eine Kopie des Projekts, an dem sie gemeinsam arbeiten sollten, allerdings wurde die Funktionalität zum Wiederverwenden des Projekts nicht genutzt.

Dies kann viele Gründe haben, die in dieser Arbeit nicht aufgeklärt werden können. Einer der Gründe könnte sein, dass die Nutzer den schnellsten Weg zum Sitzungsstart wählen, da sie dies als unangenehmsten Teil der Sitzung ansehen (Ergibt sich aus Befragungen im Anschluss an die Tests). Aufgrund der hohen Netzwerkgeschwindigkeit, und damit geringer Wartezeit bei der Übertragung der Projekte, besteht die Möglichkeit, dass die Probanden schlicht keinen Grund darin sahen, eine neue Funktionalität auszuprobieren.

Interessanterweise beschwerte sich ein Proband nach einem der Tests, das er es unübersichtlich finde, dass so viele Projektkopien bei ihm abliegen und dies im Zusammenhang mit den undeutlichen Markierungen dazu führte, dass er nicht sicher war, welches Projekt nun Bestandteil der Sitzung ist. Dies kann als weiteres Indiz dafür angesehen werden, dass die Funktionalität nicht ausreichend selbsterklärend ist.

Bei den vorherigen Tests, vor den Anpassungen des Wizards, die im Folgenden vorgestellt werden, besteht zudem die Möglichkeit, dass Nutzer schlicht übersehen haben, dass es Tabs für andere Projekte gibt. Diese Vermutung kam vor allem deshalb auf, da dies bei der eigenen Arbeit mit Saros zuvor passiert war und erst später entdeckt wurde.

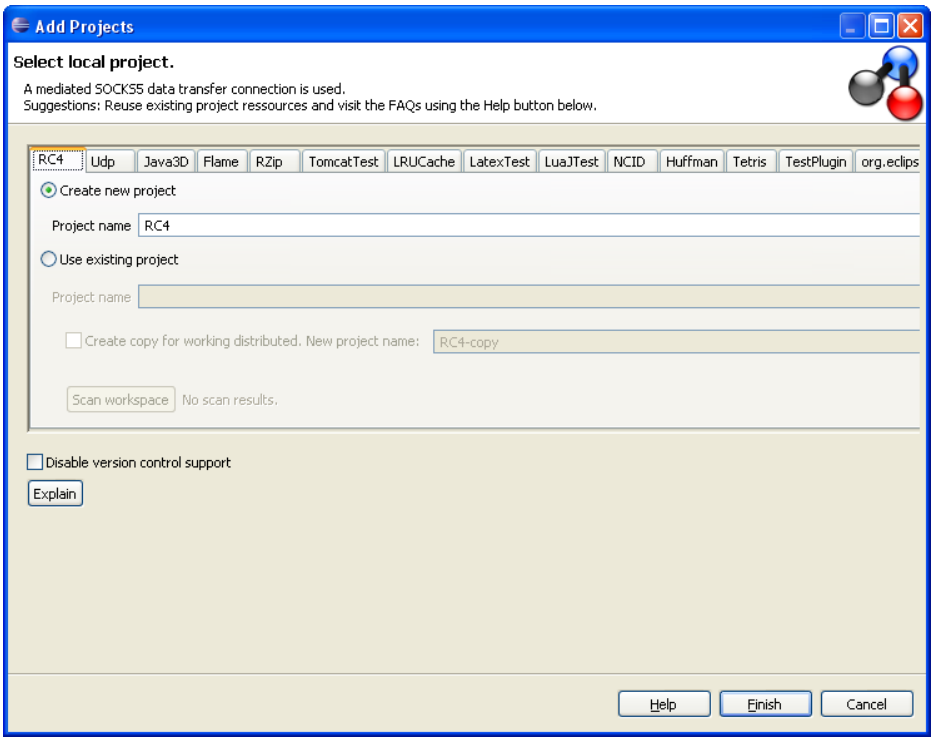

Abbildung 41: Der Wizard für eingehende Projekte nach Annahme einer Einladung

Abgesehen von der ungewohnten Darstellung der Tabs entspricht die Handhabung der Projekte im Wizard nicht den Eclipse Guidelines. Ein Wizard sollte grundsätzlich für mehrstufige Eingabeprozesse verwendet werden (Vgl. [EclipseGuideline]) und die Eingabemasken sollten für die einzelnen Projekte auch einzelne Seiten des Wizards werden. Auch die SVN-Funktionalität, insbesondere die „Explain"-Schaltfläche zur Darstellung eines (schwer lesbaren ausgegrauten) Hilfstextes, waren weder Eclipse-Guideline-Konform noch Erwartungskonform.

Das Problem bei einer Aufteilung aller Projekte auf einzelne Wizard-Seiten wäre allerdings, dass die Masken nahezu identisch aussehen würden, bis auf den Original-Namen des eingehenden Projekts. Dies stellte sich bereits bei eigenen Tests der Oberfläche als problematisch heraus. So ist schwer deutlich darzustellen, wie viele Projekte jetzt vom Nutzer bestätigt werden müssen. Die Nutzer lasen Einleitungstexte in Wizards nur dann, wenn sie ein Problem hatten, ansonsten wurde der Text ignoriert, möglicherweise, da hier für routinierte Nutzer meist nur

überflüssige Hinweistexte erscheinen. Somit wäre ein Hinweis im Kopfteil des Wizards mit großer Wahrscheinlichkeit übersehen worden.

Der Nutzer hätte bei der auf Seiten aufgeteilten Variante keine Übersicht, für wie viele Projekte der Nutzer einen Speicherort wählen muss. Die Integration einer weiteren Übersichtsseite wäre zu dem vorher erscheinenden Einladungsdialog (zur Sitzung selbst) redundant, und würde hier auch nicht weiterhelfen, da der Nutzer sich im Wizard-Prozess erinnern müsste was auf der ersten Seite des Wizards stand. Dies würde der Heuristik N6. Erkennen vor Erinnern widersprechen und zudem den Prozess verlangsamen, da der Nutzer eine weitere Seite anklicken müsste.

Weitere Abweichungen zu dieser Heuristiken wurden bei der Auswahl der existierenden Projekte entdeckt: Ein freies Eingabefeld ermöglicht es Nutzern, den Namen von Projekten frei einzutippen, wobei eine Unzahl an Fehlern auftreten kann, die kompliziert im Wizard abgefangen und gemeldet werden. So kann, abgesehen von der Eingabe eines nicht existenten Projektnamens der Nutzer auch Projektnamen so eingeben, dass dasselbe existierende Projekt für mehrere ankommende Projekte verwendet würde. Dies führt zu unzähligen Fehlern in Saros, da das System dafür nicht ausgelegt ist.

Die Eclipse Guideline 5.8 [EclipseGuideline] beschreibt hier zwar explizit, dass für die Selektion existierender Ressourcen in einem Wizard ein Browse-Button verwendet werden soll. Allerdings wird hier explizit als Beispiel angeführt, dass dies z.B. für die Auswahl von Klassen geeignet sei. Hier ist anzumerken, dass bei der Wahl von übergeordneten Klassen oder bei der Selektion von Dateien in Eclipse sehr viele Ressourcen zur Auswahl bereitstehen. Daher wird hier meist ein Browse-Dialog geöffnet, in dem auch detailliert gefiltert werden kann.

Bei der Selektion von Projekten wird auf Basis eigener Erfahrungen und Rückfragen zu anderen Entwicklern davon ausgegangen, dass meist aus einer Liste von weniger als 20 Projekten gewählt wird. Workspaces mit mehr als 20 Projekten wurden nicht beobachtet, auch nicht in dem Unternehmen, in dem die Tests durchgeführt wurden. Die Nutzungsstatistik von Saros gibt keine Informationen über den Kontext, in dem ein Saros-Anwender arbeitet. So gibt es keine Hinweise darauf, wie viele Projekte pro Einladung versandt werden, und wie viele Projekte sich im Workspace befinden. Auch konnten keine Studien gefunden werden, die Auskunft darüber geben, wie viele Projekte durchschnittlich in einem Eclipse-Workspace verwaltet werden.

Für eine kompakte Liste von Projekten bietet ein Filter-Dialog wenig Mehrwert. Dadurch, dass Saros in der Lage ist, Projekte mit identischem Projektnamen automatisch zu erkennen, muss der Nutzer bei der Wiederverwendung der Projekte eher selten etwas manuell auswählen.

Für die Überarbeitung dieses Wizards, der Bestandteil des Einladungsprozesses ist und damit mit hoher Priorität behandelt werden sollte, wurden nun Ziele formuliert die vor allem auf die Aufgaben- und Prozessangemessenheit (Vgl. [Sarodnick2006]) abzielen und deren Erreichung sicherstellen soll, dass die Bearbeitungszeit des Einladungsprozesses, seitens des Nutzers, verkürzt wird.

Ziel 1: Die Auswahl bestehender Projekte soll schneller gehen

- Schneller Wechsel für alle Projekte gleichzeitig zwischen „Projekte neu anlegen“ und „Bestehendes Projekt wiederverwenden“
- Sofortiges Bestätigen des Wizard durch geeignetes Setzen von Standard-Einstellungen möglich (Vgl. „Good defaults“ [Tidwell2011, S. 385])
    - Alle Projekte als „neue Projekte“ ablegen als Standard verhindert möglichen Datenverlust und bietet eine Schnellstartmöglichkeit

Ziel 2: Eingeladene Nutzer sollen auf einen Blick sehen, wie viele Projekte zu verwalten sind

Ziel 3: Informationen darüber, worin die Vorteile der einzelnen Methoden bestehen, stehen bereit

- Bessere Erläuterungen was passieren wird, wenn der Nutzer ein existierendes Projekt wiederverwendet
- Informationen darüber welche Vorteile die Nutzung der SVN Bindung bietet

Ziel 4: Keine unnötigen Konfigurationseinstellungen

- Die Option für das (De-)Aktivieren der SVN-Funktionalität wird nur angeboten, wenn eine SVN Bindung am eingehenden Projekt vorhanden ist

Der Prozess sollte nicht nur vereinfacht und übersichtlicher gestaltet werde, er sollte auch beschleunigt werden. So war es in der aktuellen Fassung nicht möglich für alle Projekte auf einmal zu wählen, dass die bereits existierenden Projekte wiederverwendet werden sollen. Die Erkennung der passenden Projekte über den Projektnamen funktioniert sehr gut, da die Projekte meist identisch benannt sind bei Entwicklern, die regelmäßig miteinander arbeiten, weshalb diese Möglichkeit geschaffen werden konnte durch die globale Option „Reuse Existing Projects“, das die Einstellung für alle Projekte mit nur einem Klick umstellt.

Somit kann der Wizard automatisch für alle eingehenden Projekte das lokale Projekt mit identischem Namen auswählen.

Durch die steigende Verbreitung von sogenannten Widescreen-Monitoren mit eher horizontal orientierten Auflösungen (Vgl. [WikiWidescreen]) und auch gemäß Empfehlungen aus den Diskussionsforen zur Oberflächengestaltung in Eclipse sind vertikal „lange“ Wizards zu vermeiden. Somit wurde eine horizontale Darstellung gewählt, die auch bei sehr geringen Auflösungen (z.B. 1024x768 Pixel) noch gut zu bedienen ist.

Die Standard-Einstellung des Wizards ist so gewählt, dass der Nutzer direkt mit „Finish“ bestätigen kann, und die Übertragung der Projekte sofort beginnt.

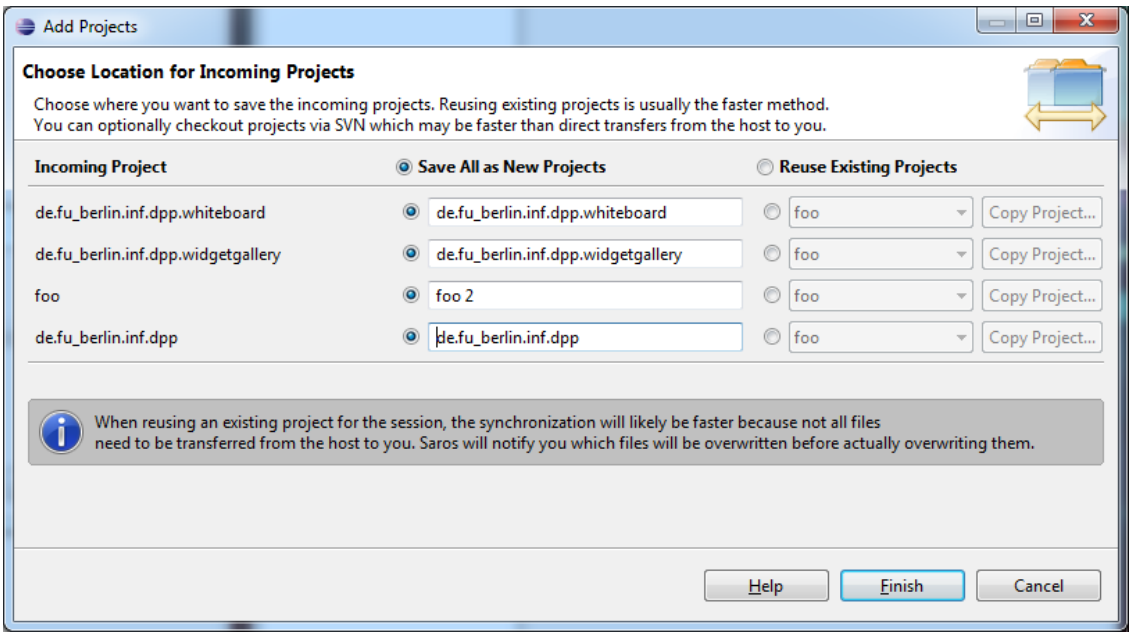

Abbildung 42: der Überarbeitete Wizard für eingehende Projekte

Die SVN-Option wird in diesem Wizard nicht angeboten, wenn keines der Projekte eine SVN Bindung hat(Heuristik N8. Ästhetisches und minimalistisches Design). Dies verkleinert den Wizard und erleichtert die Orientierung, da weniger Eingabeelemente sichtbar sind. Auf ein permanentes Darstellen der Option im ausgegrauten Zustand wurde hier bewusst verzichtet, da dies die Übersichtlichkeit des Wizards negativ beeinflussen würde.

Für eine effizientere, schnellere Abarbeitung (Heuristik N7. Flexibilität und effiziente Nutzung) der Einstellungen in diesem Wizard wurden Schaltflächen für den Schnellzugriff hinzufügt („Save All as New Projects", „Reuse Existing Projects", „UseSVN), die automatisch alle Checkboxen bzw. Radiobuttons einer Spalte umstellen.

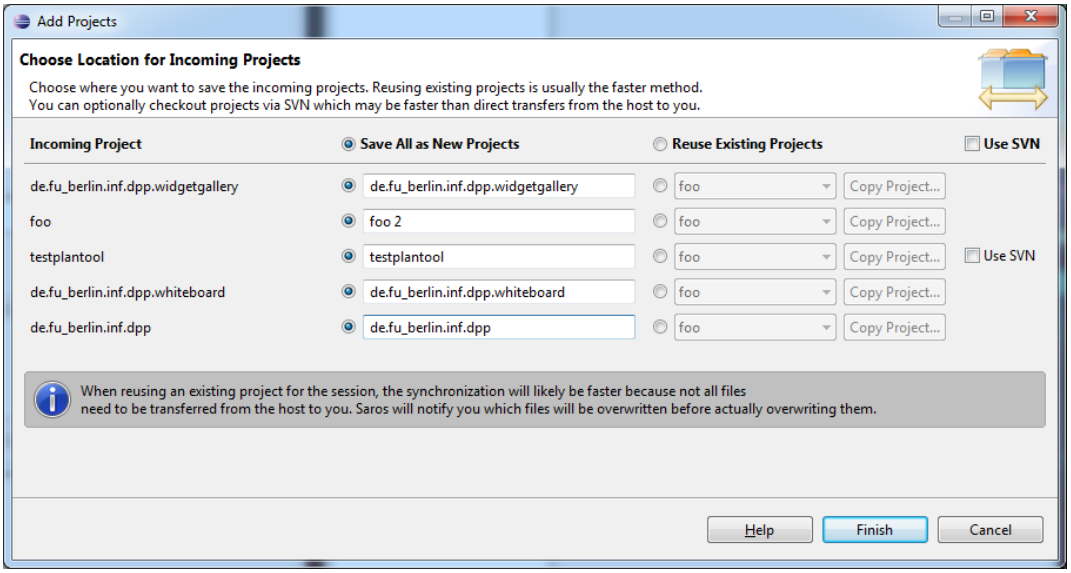

Abbildung 43: Wizard für eingehende Projekte (eines mit SVN Bindung)

So kann der Nutzer mit nur einem Klick für alle Projekte die Einstellung verändern. Er kann sowohl wählen, ob alle Projekte als neue Projekte abgelegt werden können, oder ob für alle Projekte existierende Kopien verwendet werden sollen. Auch die Nutzung der SVN-Funktionalität kann für alle Projekte mit nur einem Klick geändert werden.

Die Schnellzugriffsoptionen werden dabei auch mit den Interaktionen des Nutzers abgeglichen. Sollte ein Nutzer nur für eines der Projekte wählen, dass es ein existierendes Projekt wiederverwendet werden soll, so werden die entsprechenden Radio-Buttons im Kopf der Tabelle abgewählt. Wenn der Nutzer von Hand alle Optionen für Projekte umstellt, werden auch die Radiobuttons/Checkboxen des Tabellenkopfes umgestellt. (Heuristik N4. Konsistenz und Standards, in Anlehnung an die Eclipse-Guidelines)

Statt der freien Eingabefelder für die Projektnamen der existierenden Projekte (Verletzung der Heuristiken N5. Fehler vermeiden und N6. Erkennen vor Erinnern), wurden Auswahllisten (Dropdowns) verwendet, die ein einfaches Auswählen eines existierenden Projekts ermöglichen. Da es aus eigener Erfahrung heraus unwahrscheinlich ist, das Nutzer so viele Projekte in Ihrem Eclipse-Workspace verwalten, dass das Dropdown schlechter bedienbar wäre als ein separater Dialog, ist der ehemalige „Browse"-Dialog entfallen. Dieser war zuvor nicht erwartungskonform gestaltet, da dort auch Unterordner in einem Projekt selektiert werden konnten, und nicht nur die Projekte selbst.

Die ehemalige, verwirrende Funktionalität „Create copy for working distributed" wurde ersetzt durch eine Schaltfläche, mit der direkt eine Kopie des Projekts angelegt wird, nach der Eingabe des Projektnamens in einem kleinen Eingabedialog. Hierfür wurde ein Eclipse-Job benutzt, mit einer Fortschrittsanzeige und der Möglichkeit, den potenziell langlaufenden Prozess abzubrechen

Dadurch, dass für den Nutzer sofort zu sehen ist, dass nach dem Erstellen einer Kopie automatisch die erstellte Projektkopie zur Verwendung ausgewählt wurde, hat er die Möglichkeit, dies zu korrigieren, wenn stattdessen das Original-Projekt verwendet werden soll. Zuvor hatte der Nutzer keine Möglichkeit, dies zu erkennen, da die Kopien erst bei Bestätigen des Wizards angelegt wurden, und es in diesem Moment bereits zu spät für eine Korrektur war.

Ursprünglich war die Funktionalität „Create copy for working distributed" eigentlich dafür gedacht, dass das bestehende Projekt zur Beschleunigung der Synchronisation verwendet wird, aber ohne dass dabei das Original-Projekt verändert wird. Hier lag allerdings ein Software-Fehler in Saros vor, der dafür sorgte, dass zwar das Projekt kopiert wurde, aber keine einzige existierende Datei für die Synchronisation verwendet wurde, womit der Hauptnutzen der Funktion gar nicht gegeben war. Dieser Fehler kann nun nicht mehr auftreten, da die Kopien direkt angelegt werden, und somit für Saros wie normale Projekte behandelt werden.

5.1.5 Arbeitspaket: Präsenzinformationen

Die fehlenden Präsenzinformationen in der laufenden Sitzung von Saros, stellten eines der größten Probleme dar, weshalb hier viel Zeit investiert wurde. In diversen Iterationen wurden Skizzen eines neu gestalteten Saros-Views überarbeitet, bis schließlich eine Variante technisch umgesetzt und in Usability-Tests evaluiert wurde.

Hierbei sollte nach Gutwin und Greenberg (2004) nicht nur übermittelt werden, was ein Nutzer gerade tut, sondern auch Informationen bereitgestellt werden, die Rückschlüsse auf die Absichten eines Nutzers ermöglichen.

Anforderungen an die Überarbeitung des Saros-Views waren unter anderem, dass folgende Informationen zu erkennen sind, ohne den Arbeitsfluss störende Rückfragen zwischen den Nutzern:

- Wer welche Datei offen hat
- Wer welchem Nutzer folgt
- Wer welche Farbe in der laufenden Sitzung hat
- Wer der Host der Sitzung ist
- Wem man selbst folgt
- Wer nur Leserechte hat
- Wer welchen Dialog offen hat

Das Erreichen dieser Ziele durch den nun dargestellten überarbeiteten Saros-View konnte erfolgreich in Evaluationen mit Usability-Tests bestätigt werden, vor allem durch die Abwesenheit verbaler Übermittlung dieser Informationen zwischen Probanden. Die Übermittlung von Informationen über offene Dialoge liegt allerdings nur als Prototyp vor, da dies erst später implementiert werden konnte, und nicht mehr evaluiert wurde. Die Beschreibung dieser Funktionalität findet sich im Abschnitt der Prototypen 5.2.2.

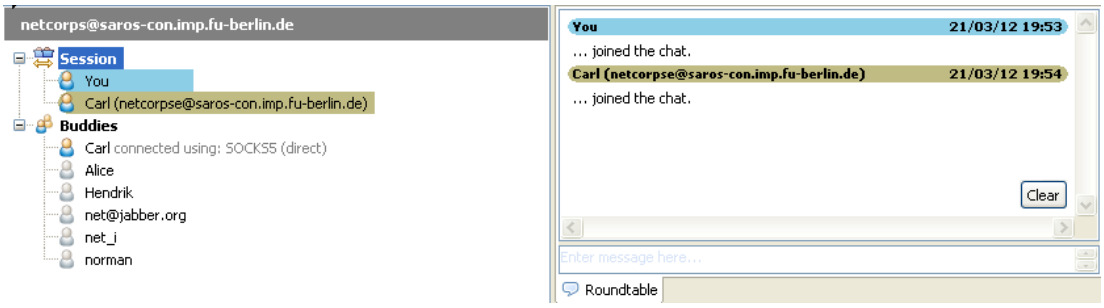

Abbildung 44: Der alte Saros-View

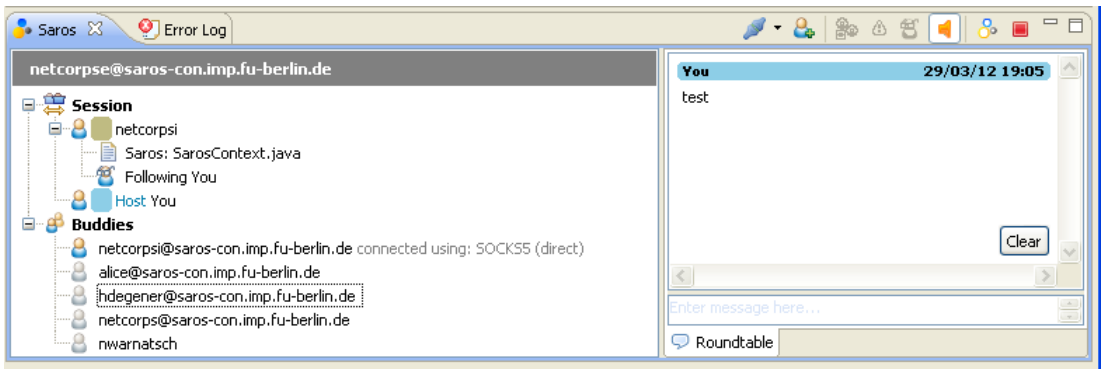

Abbildung 45: Der überarbeitete Saros-View mit mehr Awareness-Informationen

Die vollflächige farbliche Hinterlegung der Nutzer mit unsauber und unvollständig abgerundeten Ecken und Kanten wurden aufgelöst (N8 Ästhetisches und minimalistisches Design). Stattdessen werden abgerundete Farbquadrate dargestellt, neben dem Icon für den Sitzungsteilnehmer. Diese eher subtile Darstellung genügte den Probanden in der Evaluation, um ihre Farbe in der Sitzung zu erkennen. Zwar ergibt sich bei der aktuellen Implementierung das Problem, dass die Farbkachel nicht mehr sichtbar ist, wenn ein Nutzer selektiert wird, allerdings wurde das Anklicken eines Nutzers nur dann beobachtet, wenn Nutzer sich bereits in einer Sitzung befanden und sich schon über ihre automatisch zugewiesenen Farben geäußert hatten. Auch koordinierten die Nutzer zu Beginn der Sitzung, vermutlich weil noch kein größeres Vertrauen in das „neue" Saros aufgebaut werden konnte, bestimmte Informationen über den Systemzustand, darunter auch die Nutzer-Farben.

Nutzer hatten zudem die Angewohnheit, zunächst eine Datei zu öffnen und zu probieren ob diese erfolgreich synchronisiert wird. Somit war den Nutzern spätestens durch die Selektion im offenen Editor und durch die Markierungen im Saros-Chat klar, wer welche Farbe hat. Die Markierung der Farbe im Saros-View wird daher als eine geringfügige Hilfsfunktion angesehen, weshalb die deutlich subtilere Darstellung der Farben ausreicht. In der Evaluation konnten keine Probleme festgestellt werden, die auf gegenteiliges Hinweisen würden.

Zur Verbesserung in Hinsicht auf Heuristik N1 Sichtbarkeit des Systemstatus und die offensichtlich fehlenden Awareness-Informationen wurden Kindelemente hinzugefügt, die dynamisch an die Elemente der Sitzungsteilnehmer angefügt werden.

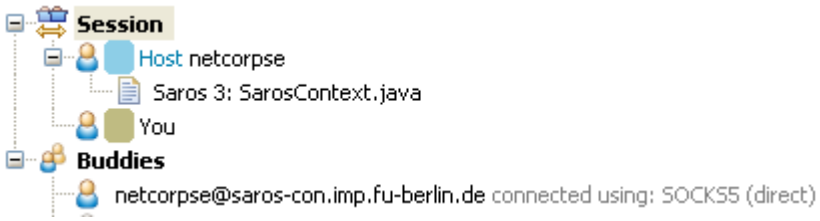

Abbildung 46: Nahaufnahme der Baumdarstellung für die Sitzung

Als festes Kindelement jedes Sitzungsteilnehmers erscheint ein Element mit Projektnamen und dem Pfad zu der Datei, die dieser Nutzer gerade aktiv offen hat. Hier wurde zunächst überlegt, ob es genügt, den Dateinamen anzugeben,

aber da beispielsweise in einem Projekt wie Saros, diverse Dateien mit identischen Dateinamen vorliegen („packge-info.java“ in jedem Java-Package), wurde die Idee verworfen. Auf das Weglassen des Projektnamens, für den Fall, dass nur ein einzelnes Projekt Bestandteil der Sitzung ist, wurde verzichtet, da dies einen zusätzlichen Hinweis beinhaltet, welches Projekt Bestandteil der Sitzung ist.

Abbildung 47: Indikation des aktiven Follow-Modus

Die Anzeige des aktiven Follow-Modus wurde verstärkt durch fette Formatierung, da die Darstellung dieses Zustands als besonders wichtig angesehen wurde. Nutzer waren sich zuvor oft unsicher, ob sie im Follow-Modus sind, daher wurde hier eine deutlichere Darstellung gewählt. Durch die Reduktion der Farbhinterlegung im Saros-View, war dieser textuelle Hinweis zudem leichter zu lesen.

Um die häufige verbale Koordination, ob der andere Nutzer im Follow-Modus ist, und z.B. ein Code-Review beginnen kann, zu verringern, wurden Saros technisch erweitert, um zusätzliche Informationen darstellen zu können.

Abbildung 48: Indikation des Follow-Modus anderer Sitzungsteilnehmer

Nutzer hatten nun die Möglichkeit auf einen Blick zu erkennen, wer wem folgt. Hierbei wurden simple textuelle Hinweise eingeblendet, und die bekannten Saros- und Eclipse-Icons („Following“ und „File“) wurden wiederverwendet.

Zusätzlich zu den Verbesserungen der Awareness-Informationen in Hinsicht auf offene Dateien und den Follow-Modus wurden kleine, aber effektvolle Änderungen vorgenommen, dadurch dass die Darstellung der Nutzeridentifikation verkürzt wurde. So erschien bei Nutzern mit einem Alias vormals die gesamte, tendenziell lange und schwer lesbare XMPP ID.

Abbildung 49: Alter Saros-View mit Alias und XMPP ID in Klammern

Die JID wird während der laufenden Sitzung nicht benötigt zur Koordination der Arbeiten. Da die Nutzer mit sehr großer Wahrscheinlichkeit wissen, mit wem sie sich gerade in einer Sitzung befinden, wurde diese Anzeige verkürzt. Wenn ein Alias gesetzt ist, wird nur noch der Alias gezeigt. Ist kein Alias gesetzt, so wird nur der Namensteil der XMPP ID angezeigt. Da es mit Saros momentan kaum möglich ist (insbesondere nicht bei Einsatz des Saros-Con-Servers) Saros-Sitzungen mit Teilnehmern unterschiedlicher XMPP Server aufzubauen, sehe ich es als überflüssig an, den Server-Teil einzublenden.

> N8. Ästhetisches und minimalistisches Design
> Irrelevante Informationen sollten möglichst vermieden werden, da hierdurch die Sichtbarkeit von relevanten Informationen gefährdet sein kann. [Nielsen1994]

Die Reihenfolge der Nutzer in einer Sitzung wurde so umgestaltet, dass das Baumelement „You“, mit dem am seltensten Interaktionen erfolgen, nun am Ende der Liste erscheint. Früher wurde der Host stets als erster Nutzer aufgelistet – die anderen Nutzer wurden dann alphabetisch sortiert. Dies scheint ein Relikt aus der Zeit zu sein, als Saros nur für 2 Nutzer konzipiert war. Die Sortierung für 3 Nutzer oder mehr wirkte auf die Anwender willkürlich, da diese keine Information darüber hatten, dass der Host stets als erstes aufgelistet wird (es gab keinen visuellen Hinweis darauf, wer der Host ist).

Die technische Rolle „Host“, die die Person kennzeichnet, die die Einladung zur Sitzung gestartet hat, und somit als einzige Person die Rechte hat, Zugriffsrechte für andere Sitzungsteilnehmer zu verändern, wird nun textuell dargestellt.

Dies widerspricht der Heuristik N2, allerdings gibt es dafür einen guten Grund.

> N2. Übereinstimmung zwischen System und realer Welt
> Das System sollte die Sprache des Benutzers sprechen. Dafür soll es keine systemorientierten Begriffe, sondern Wörter und Formulierungen, die dem Nutzer geläufig sind, nutzen. [Nielsen1994]

Zwar gibt es diverse Fehlinterpretationen der Host-Rolle durch den Nutzer, allerdings wurden die Folgen des Fehlens der Anzeige, wer der Host der Sitzung ist, als schwerwiegender gewertet, als mögliche Fehlinterpretationen des Begriffs.

Nutzer waren sich in mehreren Usability-Tests unklar, wer der Host ist und mussten durch Probieren herausfinden, wer die Rechte in der Sitzung ändern kann. Durch das Einführen der Anzeige, wer der Host ist, hatten die Nutzer nun eine Darstellung der besonderen Rolle des markierten Nutzers und arbeiteten somit schneller, da keine Rückfrage mehr notwendig war.

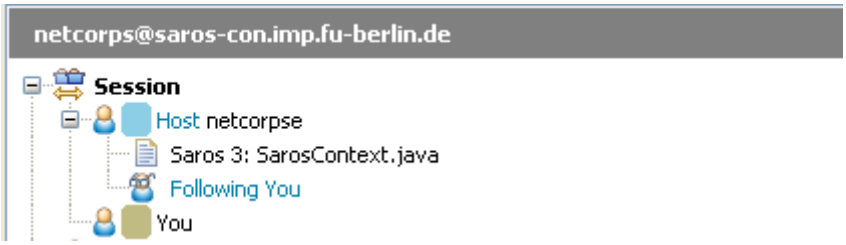

Abbildung 50: Indikation der HOST-Rolle für den Nutzer „netcorpse“

Langfristig soll die Host-Rolle abgeschafft werden, da Saros ein vollkommen gleichberechtigtes Peer-to-Peer-System werden soll. Leider ist dies nicht in absehbarer Zeit zu realisieren, und es ist den Diskussionen im Saros-Team zufolge wahrscheinlicher, dass es dazu kommen wird, dass Nutzer die Host-Rolle an andere Nutzer übertragen können, wodurch die Bedeutsamkeit der Rolle noch weiter gefestigt würde.

Ein möglicher Lösungsansatz wäre es, alle Nutzer gleichberechtigt zu schalten, sodass jeder die Rechte jedes Nutzers verändern kann. Dies würde allerdings die „Rechte“ auf ein „Readonly zur Sicherheit gegen versehentliche Änderungen“ reduzieren. Dies dürfte für Entwickler schwer zu verstehen sein, daher muss hier noch einiges an Zeit investiert werden in weiteren Untersuchungen, um eine geeignete Lösung zu finden.

Nutzer hatten Schwierigkeiten, zu ermitteln, welche Projekte Bestandteil der Sitzung sind. Die verwendeten Icon-Markierungen stellten sich als ungeeignet heraus, weshalb zusätzlich ein textueller Suffix an die Projektnamen in den Baumdarstellungen hinzugefügt wurde.

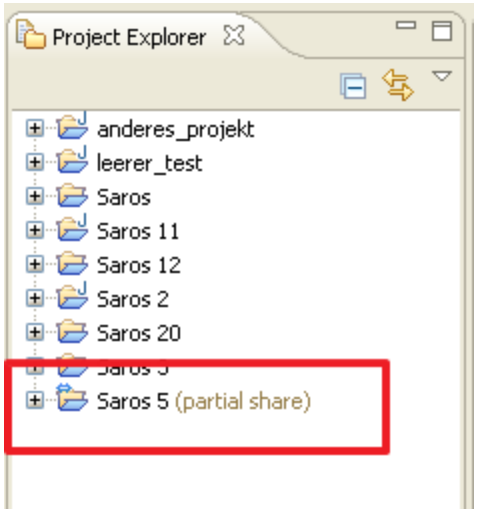

Abbildung 51: Markierung eines partiell geteilten Projekts

Hierbei wird unterschieden zwischen „(partial share)“ für partiell geteilte Projekte, und „(shared)“ für komplett geteilte Projekte. Diese kleine Änderung wurde von den Probanden sofort positiv aufgenommen und verringerte die Dauer für die Überprüfung, ob ein Projekt erfolgreich in die Sitzung aufgenommen wurde deutlich. Den Probanden genügte beim Prototypen ein einziger Blick auf den Package Explorer während die Nutzer zuvor im Package Explorer herumklickten, um zu erkunden ob die Inhalte des Projekts tatsächlich Bestandteil der Sitzung sind.

5.1.6 Arbeitspaket: Angleichung der Ballon-Benachrichtigungen

Die sogenannten Ballon-Benachrichtigungen wurden überarbeitet, um in Ihrer Ästhetik eher der bekannten Darstellung von Windows zu folgen und sie wurden in ihrem Verhalten angeglichen.

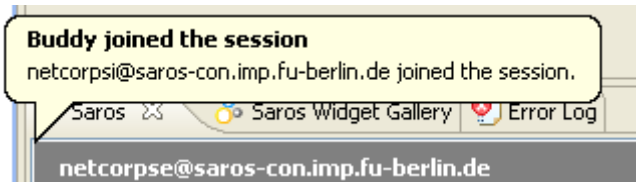

Abbildung 52: Ballon-Benachrichtigung für das Beitreten eines Nutzers zur Sitzung

Neben der Vereinheitlichung der Hintergrundfarbe dient nun die gesamte dargestellte Fläche der Benachrichtigung als Schließen-Schaltfläche. Leider war es nicht möglich, das Schließen-Kreuz in der rechten oberen Ecke weiterhin einzublenden, da bei Verwendung der dafür zuständigen Einstellung SWT.CLOSE keine Lösung gefunden wurde, um das Schließen des Dialogs auch bei Klick auf den Text oder den Hintergrund auszulösen. Hier können weitere Verbesserungen vorgenommen werden.

Neben der Bereinigung der Darstellung wurde das Verhalten der Benachrichtigungen in vielen Situationen verbessert.

So ersetzt eine Benachrichtigung etwaige noch existierende Benachrichtigungen. Es gibt zu jedem Zeitpunkt nur eine offene Benachrichtigung, was dem Verhalten von Windows-Ballons entspricht. Dies erübrigt das manuelle Schließen mehrerer offenen Benachrichtigungen. Gerade beim (versehentlichen) Editieren als READONLY Nutzer wurde hier jeweils eine Benachrichtigung pro Tastendruck ausgelöst.

Auch werden bei Sitzungsende nun Benachrichtigungen ausgeblendet, die sonst noch bis zu 15 Sekunden lang bestehen bleiben würden und in den Beobachtungen die Nutzer deutlich gestört hatten. Wenn beispielsweise ein Teilnehmer erst den Follow-Modus verlässt, und danach die Sitzung endet (eine Art „Aufräumen“-Verhalten der Probanden, das in Tests beobachtet wurde) blieb die Benachrichtigung dafür früher noch lange bestehen. Diese wird nun sofort ausgeblendet, was auf Nachfrage bei den Probanden, nach den Tests eines Prototypen, für ein deutlich „besseres Gefühl“ sorgte, da das Ende einer Sitzung kein „Zeug hinterlässt“.

5.1.7 Arbeitspaket: Einführung in Saros (Tutorial)

Der alte Getting Started Wizard, der gegen die Eclipse Guidelines und diverse Gestaltungsrichtlinien verstößt, wurde entfernt.

Hier gab es mehrere Optionen für eine Überarbeitung:

- Erstellen von Seiten innerhalb der Eclipse-Hilfe
    - Die Eclipse-Hilfe wird als schmales Fenster an der Seite der Perspektive dargestellt, und ist damit für große Screenshots oder längere Erläuterungen ungeeignet

- Einbettung in den „Eclipse Welcome Screen“
    - Der Welcome Screen wird aus eigener Erfahrung von vielen Nutzern sofort weggeklickt und nicht als ernst zu nehmende Informationsquelle angesehen. Zudem wäre die Umsetzung aufwändig.

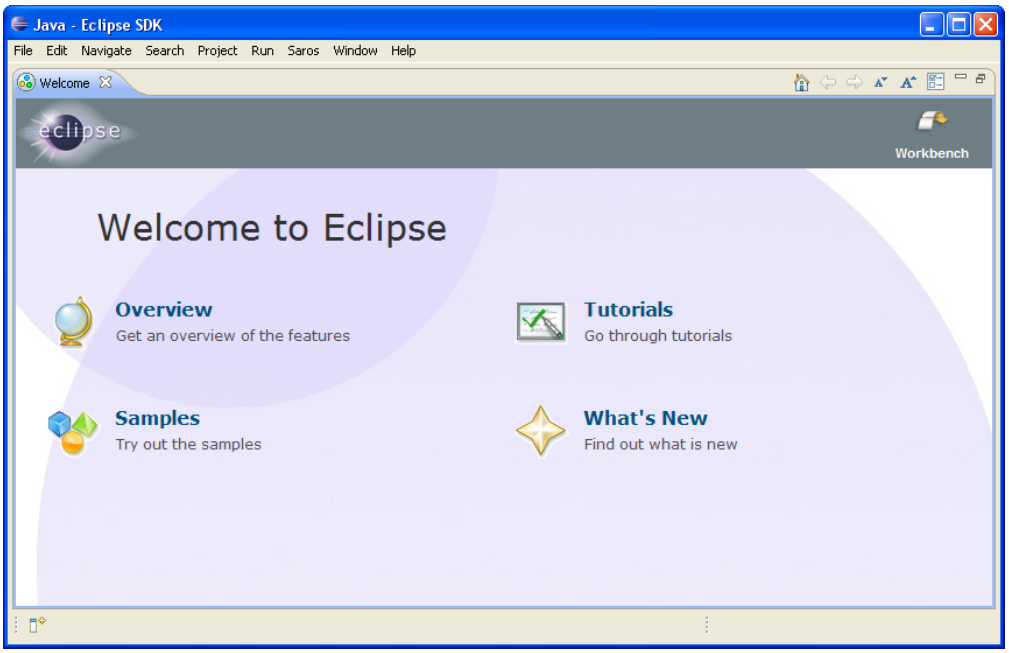

Abbildung 53: Der Eclipse Welcome Screen

Stattdessen wurde entschieden, dass innerhalb von Eclipse nun ein Eclipse-interner Webbrowser mit der Saros-Webseite geöffnet wird, in der das alte Tutorial integriert wurde. Eine Verbesserung des Tutorial-Inhalts an sich steht noch aus – allerdings ist gerade eine Bachelorarbeit, zur Überarbeitung der gesamten Webseite, ausgeschrieben. Daher gehe ich davon aus, dass dies bald erfolgen wird. Ein Link auf den Funktionsumfang von Saros wurde bereits in die Tutorial-Webseite integriert, um es Nutzern zu ermöglichen, sich nach der Einrichtung über alle Funktionen von Saros zu informieren.

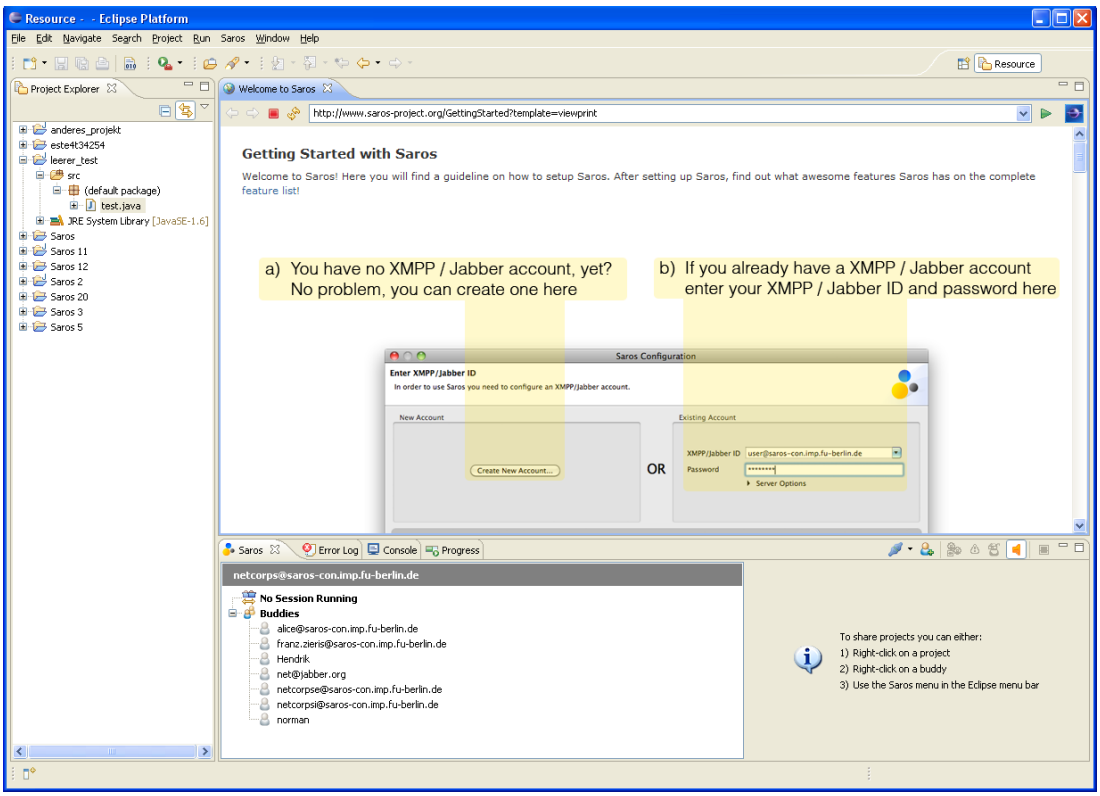

Abbildung 54: Das neue Getting Started in Eclipse

Vorteile dieser Lösung sind vor allem die folgenden:

- Eine Aktualisierung/Verbesserung des Tutorials ist unabhängig vom Saros-Release möglich
- Das Tutorial kann interaktiver gestaltet werden (mit HTML, CSS, Javascript)
- Videos können eingebettet werden
- Verlinkung / Integration der Feature-Liste ist möglich (und auch vorhanden)
    - Dies fördert die Erlernbarkeit durch besseres informieren über den Funktionsumfang
- Leichtere Gestaltung als zuvor (Wiki-Syntax zur Bearbeitung der Webseite, vor allem für SWT-Laien praktisch)
- Betriebssystemabhängige Screenshots können eingeblendet werden (momentan: MAC-Screenshots)
    - Es könnte Nutzer verwirren, wenn sie Screenshots im Mac-Stil sehen, aber ein Windows-Betriebssystem nutzen

Probleme damit, dass die Webseite von Saros nicht erreichbar ist, aufgrund von Proxy-Einstellungen oder Ähnlichem sollten so selten sein, das dies vernachlässigbar ist für eine „Einführung“, die nicht Hauptbestandteil der Dokumentation von Saros ist.

5.1.8 Arbeitspaket: Dashboard

Um zu verhindern, dass dem Nutzer gleich zwei modale Wizards angezeigt werden (Getting Started & Konfiguration), was zu Frustration bei den Nutzern führte, wurde ein Dashboard eingeführt. Dies ist ein Tab der immer angezeigt wird, und dem Nutzer somit die Gelegenheit gibt, zu erkennen, dass im Saros-View ein

Tab-Control verwendet wird, und sich dort vermutlich noch weitere Tabs öffnen werden.

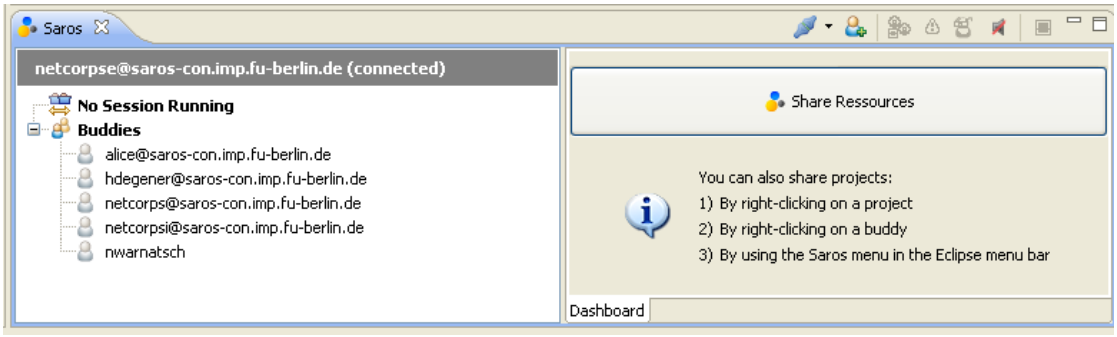

Abbildung 55: Das Dashboard im Saros-View

Das Dashboard hat grundsätzlich folgende Funktionen:

- Schaltfläche zum Öffnen der Konfiguration wenn der Nutzer die Konfiguration noch nicht abgeschlossen hatte
    - Dies erspart das automatische Öffnen des Konfigurations-Wizards beim erstmaligen Start von Saros (bzw. beim Wechseln von Eclipse-Workspaces)
- Schaltfläche zum Öffnen des Wizards für die Einladung zur Sitzung
    - Diese Schaltfläche wurde in den Tests von Nutzern sehr positiv aufgenommen, es wurde fast ausschließlich über die Schaltfläche gestartet, nicht mehr über Kontextmenüs
    - Spart einen Klick im Vergleich zu den Kontextmenüs bei Selektion mehrerer Projekte
    - Ist ein guter Startpunkt für ungeübte Nutzer, die Kontextmenü und Saros-Menü nicht so schnell entdecken
- Darstellen von zusätzlichen Informationen möglich

Dieses Dashboard kann noch um folgende Informationen ergänzt werden:

- Liste der letzten Saros-Sitzungen (per Doppelklick kann die Sitzung gestartet werden)
- Knopf zum erneuten Starten der letzten Sitzung durch einen Klick

Die technischen Voraussetzungen wurden dafür bereits geschaffen. So speichert Saros automatisch die letzten Sitzungsteilnehmer (im Einladungsprozess) und die letzte Ressourcen-Selektion im Einladungsprozess.

Ein Prototyp der Funktionalität für das „Session Resuming“ liegt im Saros-Reviewboard, ist allerdings noch nicht einsetzbar, und konnte noch nicht vollständig evaluiert werden.

5.1.9 Arbeitspaket: Konsistenz und Standards

In diesem Arbeitspaket wurden einige deutlich bemerkbare, aber eher kleinere Anpassungen vorgenommen, um die Oberfläche von Saros in Bezug auf die Eclipse Guidelines und etablierte Gestaltungsrichtlinien anzugleichen.

Die Arbeit an der Behebung von Usability Problemen birgt Frustrationspotential. So kommt es in Usability Evaluationen recht häufig dazu, dass Nutzer komplett anders denken und handeln, als man es angenommen hatte, und nicht mit der neuen, persönlich als optimal empfundenen, Oberfläche zu Recht kommen. Auch wusste ich bereit zu Beginn der Arbeit, dass ich nicht in der Lage bin, 6 Monate ohne Unterbrechung, nur die Aufgaben mit der höchsten Priorität zu behandeln, die zumeist auch die am Schwierigsten zu lösenden Probleme darstellen. So wurden die notwendigen, und teilweise ausbleibenden Erfolgserlebnisse, zur Hebung meiner eigenen Moral dadurch herbeigeführt, dass kleinere Verbesserungen von geringer gewichteten Problemen vorgenommen wurden, die dennoch einen bedeutsamen Teil zum „Look & Feel" Aspekt von Saros beitragen können.

Bei Abgleich mit den Eclipse Guidelines (7.16 [EclipseGuideline]) fällt auf, dass Buddies, die Bestandteil einer Sitzung sind, doppelt angezeigt werden. Sowohl in der Liste der Buddies, als auch in der Liste der Sitzungsteilnehmer. Dies führte unter anderem dazu, dass Nutzer längere Zeit brauchten, um herauszufinden, wie sie einem Nutzer folgen können, da zunächst das „falsche", abweichende Kontextmenü des Buddies angeklickt wurde, und nicht das Kontextmenü des Buddies in der Liste der Sitzungsteilnehmer.

Hierzu wurde ein *ViewerFilter* für die Baum-Darstellung implementiert, die künftig Nutzer aus der Buddyliste ausblendet, die bereits in der Liste der Sitzungsteilnehmer aufgeführt werden.

Bei dieser Änderung geht nun allerdings eine sehr technische Information verloren: die Art des Verbindungstyps. Diese Angabe widersprach der Heuristik N2 Übereinstimmung zwischen System und realer Welt, und sollte vermutlich der Heuristik N10 Hilfe und Dokumentation bzw. N9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern, entsprechen. Das Problem bei der gewählten Darstellung war allerdings, dass sich mehrere Nutzer über die Saros-Umfrage dahingehend äußerten, dass sie nicht in Lage seien zu verstehen, was der Verbindungtyp bedeutet (z.B. „connected via IBB") und wie sie eine bessere Verbindung herstellen können, ohne zunächst eine Schulung zum Netzwerktechniker absolvieren zu müssen.

Dass diese Information nicht mehr an dieser Stelle dargestellt ist, stellt somit kein Problem dar, solange sichergestellt wird, dass eine bessere Hilfe angeboten wird, die den Nutzern erklärt, was die einzelnen Verbindungstypen bedeuten, und was sie unternehmen können, um das Problem (z.B. einer langsamen IBB-Verbindung) zu beheben. Da die Zeit nicht mehr reichte, um diese Anpassungen an der Webseite und dem Saros-View vorzunehmen, wurde diese Änderung zwar in das Saros-Review-Board hochgeladen, muss allerdings dort verweilen, bis eine geeignete Lösung für die Hilfestellung bei Verbindungsproblemen gefunden wurde.

Bei der Betrachtung der Kontextmenüeinträge, z.B. auf den Buddies im Saros-View fiel auf, dass der am häufigsten genutzte Eintrag „Work together on > …." sich ganz am Ende des Kontextmenüs befand. Dies führte zu zusätzlichen „Mauskilometern" (Mausstrecke die auf dem Bildschirm zurückgelegt werden muss), und verringerte die Effizienz bei der Arbeit mit Saros. Diese Kontextmenüs verstoßen gegen die Heuristik N4 Konsistenz und Standards, da die Eclipse

Guideline 7.14 angibt, dass die Standard-Reihenfolge für Menüeinträge verwendet werden soll [EclipseGuidline].

Gemäß der [EclipseGuideline] und der [MicrosoftGuideline], für die erwartungskonforme Gestaltung der Oberflächen, wurden die Kontextmenüeinträge in der Benennung vereinheitlicht und umsortiert.

New
Open
Navigate + Show In
Cut, Copy, Paste, Delete, Move, Rename and other refactoring commands
Other Plugin Additions
Properties

Abbildung 56: Vorgegebene Kontextmenü-Reihenfolge aus den Eclipse-Guidelines [EclipseGuideline]

Die Gruppierung der Einträge war soweit bereits korrekt erfolgt, lediglich die Reihenfolge entsprach nicht einer logischen Abfolge und auch nicht der Nutzungshäufigkeit. So erschien beispielsweise bei einem Buddy der Eintrag „Delete", zum Löschen des Buddies aus der Kontaktliste vor dem „Work together on > …" zum Starten einer neuen Sitzung.

Die Aktion „Work together on > …" zum Starten einer neuen Sitzung passt am ehesten in die Kategorie „New", und wird am häufigsten verwendet. Diese wird somit als erster Menüeintrag eingeblendet.

Hierbei wurde in Kauf genommen, dass die Nutzer, die Saros bereits kennen, zunächst stutzen werden, da der Menüeintrag sich nicht mehr am Ende des Menüs befindet. Dies kann kurzzeitig zu einer Verzögerung in der Bedienung führen, sollte langfristig aber eine Verbesserung darstellen.

Um die Orientierung im Kontextmenü weiter zu erleichtern und die Ästhetik zu verbessern, wurden allen Menüeinträgen in den Kontextmenüs Icons hinzugefügt, die aus der Liste der Eclipse-Standard-Icons entnommen wurde. Nach Wiedenbeck (1999) helfen Icons signifikant dabei, sich in einer neuen Oberfläche erstmalig zu Recht zu finden. Auch tragen sie dazu bei, den „ease of use" zu verbessern, und verbessern die User Experience deutlich.

Zusätzlich wurden Menüeinträge einheitlich bezeichnet, da diese in der Benennung teilweise stark unterschiedlich waren.

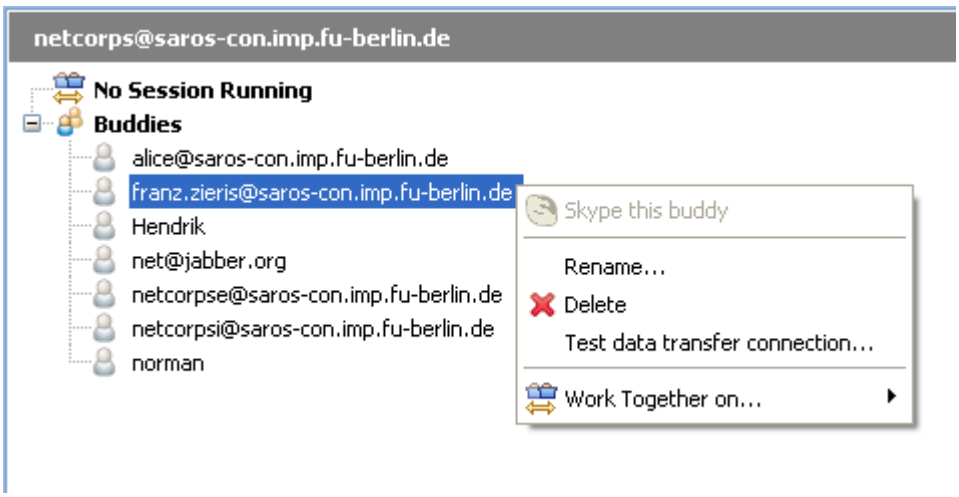

Abbildung 57: Kontextmenüs vor der Überarbeitung

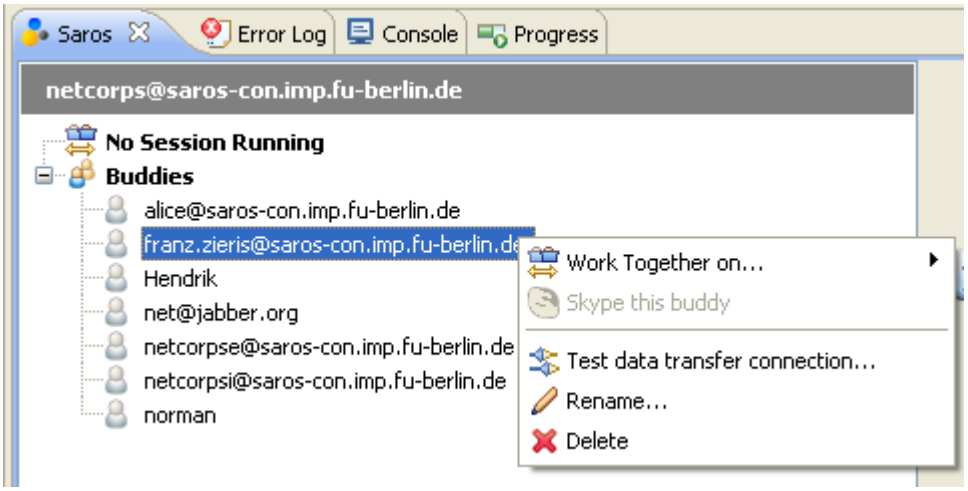

Abbildung 58: Kontextmenüs nach der Überarbeitung

Für die Funktion „Test data transfer connection“ wurde vorgeschlagen diese zu erweitern, sodass die Funktion eine detaillierte Hilfe einblendet (ähnlich dem Saros-Tutorial). Die Sinnhaftigkeit und die Anforderungen an die schwache „Skype-Integration“ wurde im Bugtracker hinterfragt.

Eine weitere erwähnenswerte Veränderung, war das Speichern der Aufteilung des Saros-Views. So mussten Probanden wiederholt den Chat verkleinern, um alle Awarenessinformationen im Saros-View zu sehen, da der Chat initial 50% des verfügbaren Platzes beansprucht, auch wenn dieser nicht zum Einsatz kommt. Es gibt keine Möglichkeit den Chat auszublenden. Abhilfe schaffte hier das wiederherstellen der letzten, durch den Benutzer vorgenommenen, Aufteilung.

5.2 Prototyp: Changelog für Saros-Sitzungen

Für den Host der Session (und auch für andere Teilnehmer der Session) gab es keine Möglichkeit, herauszufinden, wer zu welchem Zeitpunkt innerhalb einer Saros Session Dateien modifiziert hat.

Diese Information ist wichtig, insbesondere wenn es darum geht zu ermitteln, wer für eine potenziell fehlerhafte Code-Veränderung verantwortlich ist. Bei Einsatz

einer Versionsverwaltung wie SVN, gibt es eine Indikation durch ein kleines Symbol am Icon der Datei im Package Explorer, dass anzeigt, dass die Datei modifiziert wurde, gegenüber der letzten SVN-Revision. Allerdings ist es nach einem Commit des Session Hosts nicht mehr möglich zu ermitteln, ob die Datei gemeinsam bearbeitet wurde, oder nur durch die Person die die Änderungen via SVN commit veröffentlicht hat.

Es gibt lediglich eine Indikation, beispielsweise im Package Explorer, in Form von Icon-Annotations, die deutlich machen, welche Dateien gerade geöffnet sind und welche Dateien im Hintergrund als Tabs in Eclipse offen sind. Diese gibt aber keine Information darüber, ob die Datei tatsächlich modifiziert wurde oder gerade modifiziert wird. Hier wäre anzudenken, eine alternative Icon-Annotation einzuführen, die indiziert, das gerade aktiv in einer Datei gearbeitet wird.

Bei der Einarbeitung in den Code von Saros wurde, weniger zur Behebung des Problems, vielmehr zum Sammeln von Erfahrungen, im Umgang mit den von Saros bereitgestellten Schnittstellen, ein „Changelog" prototypisch entwickelt. In einer laufenden Sitzung wird bei jedem Dateizugriff erfasst, welcher Nutzer innerhalb einer Datei editiert hat. Hierbei wird zunächst nur erfasst, dass eine Datei durch einen Nutzer editiert wurde, aber nicht, ob diese tatsächlich verändert wurden. (Durch Eingabe & Rückgängig machen der Eingabe wäre die Datei editiert, aber unverändert).

Hierzu wurde ein *Listener* an den *EditorManager* von Saros gehängt, der mitprotokolliert wenn ein bestimmtes Ereignis eingetreten ist (*textEditReceived*)

```
editorManager
    .addSharedEditorListener(new AbstractSharedEditorListener() {
        @Override
        public void textEditRecieved(User user, SPath file,
            String text, String replacedText, int offset) {
            changelog.rememberTouch(user, file);
        }
    });
```

Abbildung 59: Listener für das Changelog

Die erstellte Changelog-Datei beinhaltet dabei eine Liste von modifizierten Ressourcen (und auch nur denen) mit einer Liste der XMPP IDs der Nutzer, die diese Datei editiert hatten.

5.2.1 Prototyp: Sitzungs-Neustart mit einem Mausklick

Nutzer äußerten sich bei Befragungen und in Usability-Tests dahingehend, dass ein schnelleres „Fortsetzen" der letzten aktiven Saros-Sitzung wünschenswert wäre für eine effizientere Arbeit mit Saros. Hierzu wurden bereits erste Anpassungen durchgeführt, die dies erleichtern. So können Nutzer komplexe Ressourcen-Selektionen unter frei wählbaren Namen abspeichern und wiederherstellen (bereits in Abschnitt 5.1.2 erwähnt) und Saros speichert automatisch die jeweils letzte Selektion die bei der Sitzungseinladung verwendet wurde.

Hierbei folge ich dem Ansatz „Streamlined Repetition" [Tidwell2011], bei dem oft wiederholte, gleichartige Aktionen, für den Nutzer auf wenige Klicks reduziert werden sollten. Wünschenswert wäre hier also noch, dass Nutzer direkt mit einem

Klick den Einladungsprozess starten können, ohne noch etwas selektieren zu müssen.

So wurde die neu eingeführte *SelectionsStore* Klasse zur Ablage der Selektionen in den Eclipse Preferences, um Funktionen zum Speichern der zuletzt eingeladenen Buddies (die XMPP IDs werden hierzu gespeichert) erweitert. Die *CollaborationUtils* Klasse, über die Einladungen initiiert werden, wurde so angepasst, dass diese automatisch die zuletzt eingeladenen Buddies speichert.

Hierdurch wäre es nun möglich eine noch einzuführende Schaltfläche (auch ein Icon in der Toolbar des Saros-Views wäre denkbar) für das „Session Resuming" einzubinden, mit der Nutzer die letzte Sitzung fortsetzen können.

Hierbei existieren aber noch einige konzeptionelle Probleme die zeitlich nicht mehr gelöst werden konnten:

- So wird nur die initiale Selektion bei der Einladung gespeichert. Nachträglich zur Sitzung hinzugefügte Ressourcen wären somit nicht Bestandteil der Sitzung. Dies würde für Nutzer wahrscheinlich nicht erwartungskonform sein.
- Auch ist es schwierig einzuschätzen, ob Nutzer erwarten, dass nachträglich zur Sitzung hinzugefügte Benutzer beim Fortsetzen der Sitzung automatisch mit eingeladen werden sollen. Leider ist es schwer zu bestimmen, wann und wie die Sitzungsteilnehmer für das Fortsetzen gespeichert werden sollen, um möglichst Erwartungskonform zu sein. So wäre es auch abweichend von der Nutzererwartung, wenn eine Person, die sich nur 10 Sekunden in der Sitzung befand, und diese sofort wieder verlassen hat, beim Fortsetzen der Sitzung mit eingeladen würde.
- Es wäre auch nicht unproblematisch, die letzten Sitzungsteilnehmer erst zum Ende der Sitzung zu speichern, da es sein kann, dass Nutzer die Sitzung vor dem Host verlassen, bevor die Sitzung endet. In diesem Fall kann es sogar sein, dass der Host alleine in der Sitzung verbleibt und ein Fortsetzen der Sitzung somit sinn frei wäre.
- Der Host kann sich womöglich nicht erinnern, was genau passieren wird, wenn er die Schaltfläche zum Fortsetzen der letzten Sitzung betätigt (Widerspräche Heuristik N6. Erkennen vor Erinnern).

Daher wurde weiter konzipiert: Anstelle der Hinweise, wie eine Sitzung gestartet werden kann, könnte eine Liste der letzten Sitzungen dargestellt werden, die jeweils per Doppelklick auf einen Eintrag fortgesetzt werden können.

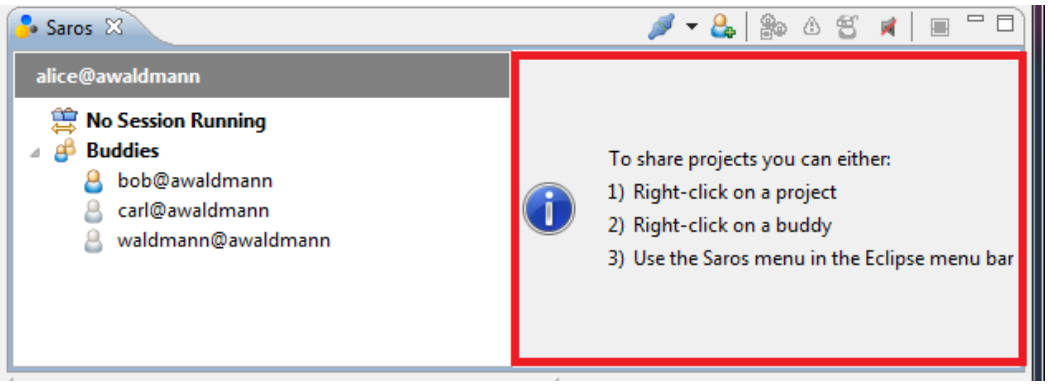

Abbildung 60: Der Saros-View: Hier wäre Platz für die Sitzungshistorie

So hätten Nutzer eine Übersicht welche Sitzungen kürzlich stattfanden und könnten diese schnell fortsetzen. Allerdings konnte hier noch kein Konsens gefunden werden, wie die Sitzungen aufgelistet werden sollen, um dem Nutzer zu verdeutlichen, welche Sitzung es war. Denkbare Attribute einer Sitzung die dargestellt werden könnten wären

- Startzeitpunkt
- Endzeitpunkt
- Liste von Teilnehmern (Anzeigenamen oder XMPP IDs)
- Liste der Projekte
- Eigener Sitzungs-Name der durch Nutzer vergeben wird

Aufgrund von geringem Platzangebot im Saros-View, ist es schwierig, hier zu entscheiden, welche Informationen wichtig wären. Eine weitergehende Konzeption dieser Funktionalität konnte aus Zeitgründen nicht mehr vorgenommen werden – somit verbleibt es hier bei der Implementierung der technischen Voraussetzungen und einem ersten Lösungsansatz.

5.2.2 Prototyp: Präsenzinformationen

Die Anwender hatten bei der Nutzung von Saros Probleme damit, dass Saros nicht übermittelt, ob und welchen Dialog ein Nutzer öffnet (Absichten der Nutzer, Vgl. [Gutwin1996], [Gutwin2004]).

So gab es mehrfach Rückfragen der Probanden nach dem „was tust du gerade?“, wenn sich in der Ansicht von Eclipse nichts veränderte in einer laufenden Sitzung. Dies trat vor allem auf, wenn ein Nutzer die Suchfunktion verwendete („Search“) oder eine neue Java-Klasse anlegte („New Java Class“).

Um diese Informationen zu übermitteln, wurden weitere *Activity*-Klassen eingeführt, und entsprechende *Listener* am Eclipse Workspace angehängt, die überwachen, welche Fenster ein Nutzer offen hat, und bei neuen Fenstern (Dialogen) den Titel der Fenster an alle Sitzungsteilnehmer weiterleiten. Auch das Schließen von Fenstern wird somit übertragen.

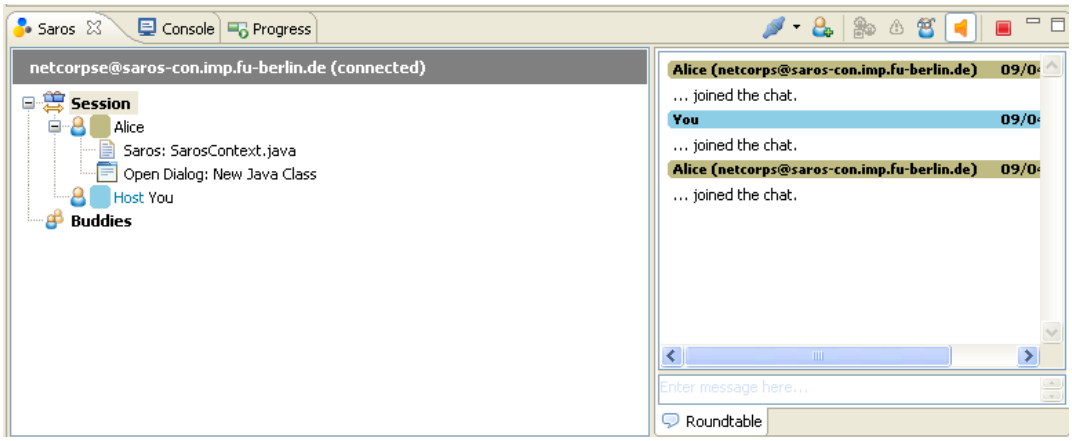

Abbildung 61: Verbesserung der Awareness durch Übermittlung von Dialogtiteln

Da diese Funktionalität nicht mehr vollständig implementiert werden konnte (nicht alle Randbedingungen wurden abgedeckt), und auch nicht in Tests mit Nutzern evaluiert wurde, verbleibt es im Review-Board von Saros, bis die entsprechenden Arbeiten abgeschlossen sind und die Funktionalität sich im Praxiseinsatz beweisen konnte.

5.3 Ergebnisse der Evaluation der Überarbeitung

Nachdem in den Arbeitspaketen vorgestellt wurde, welche Änderungen in Saros durchgeführt wurden, wird in diesem Abschnitt betrachtet, was die Evaluation der Prototypen ergab und wie sich die Aussagen der Probanden in den Fragebögen veränderten.

Die Probanden wirkten in den Evaluationen deutlich sicherer, dass Ihre Interaktion mit dem System einen Effekt hatte, und es auch der gewünschte Effekt war. Während bei der Evaluation des Original-Saros keine positiven Äußerungen der Nutzer beobachtet wurden, äußerten sich in der Evaluation der Prototypen einige Nutzer positiv hin Hinsicht auf schnelle Rückmeldungen durch das System.

Bei den Evaluationen erfolgte weniger verbale Koordination durch die Probanden. So wurde nicht mehr danach gefragt, welche Datei jemand offen hat – vielmehr wurde nun von den Probanden vorgelesen, welche Datei der andere Nutzer offen hat, nach dem die Probanden diese neue Funktionalität im Saros-View entdeckten, und positiv kommentierten. Diese Koordination erfolgte möglicherweise, um zunächst sicherzugehen, dass die Information tatsächlich korrekt ist, da bei der Nutzung der vorherigen Version von Saros zuvor wenig Vertrauen aufgebaut werden konnte.

Eine Vielzahl der durchgeführten Anpassungen aus Arbeitspaket 1 (Verbesserung der Performance und des Feedbacks) waren bereits im März Release enthalten. Diese konnten auch bei einem Akzeptanztest von anderen Saros-Nutzern getestet werden. Hier gab es sowohl positives wie auch negatives Feedback. Die Ergebnisse des Akzeptanztests fielen gemischt aus: Die Menge und der Detailgrad an Feedback durch das System wurde als sehr positiv aufgenommen. Der Saros-View wurde jedoch als uneinheitlich bezeichnet. Bei diesen Tests war die Indikation, wer wem folgt, noch in einer anderen Schriftfarbe dargestellt, was als inkonsistent aufgefasst wurde, aber schnell behoben werden konnte.

Abbildung 62: Screenshot aus der Durchführung des Akzeptanztests

Zudem bemängelten die Tester, dass die Indikation, wem man selbst folgt nicht einheitlich sei. Im Screenshot folgt der aktuelle Nutzer, der den Screenshot erstellt hat, dem Nutzer Stephan. Dies ist dargestellt durch den Text (following) hinter dem Nutzernamen. Dies war bereits in früheren Saros-Versionen der Fall. Für die

Anzeige des Follow-Modus anderer Nutzer wird für das Baumelement des Nutzers ein Kindelement im Baum integriert. Dieses verwendet das Icon für den Follow-Modus um eine eindeutige Zuordnung zu erzielen.

Denkbar wäre hier gewesen, auch für den Eintrag „You“ (für den aktuellen Nutzer) mit einem Kind-Element auszustatten (dies geht durch auskommentieren einer einzelnen Zeile Java-Code). Das Problem bei der Variante war allerdings, dass, diese Darstellung zu komplex wirkte in eigenen Tests des Prototypen (schon bei 3 Nutzern in der Sitzung wird der Platz im Saros-View, in der üblichen Größenordnung im Eclipse Fenster knapp). Somit wurde hier die zuvor verwendete Darstellung des Follow-Modus verwendet, und erweitert, um Kontextinformationen über die Aktivitäten der anderen Nutzer.

Es bleibt somit offen, in weiteren Usability Tests zu ermitteln, ob der Saros-Views noch weiter verbessert werden muss, und wie dies am besten erfolgen sollte. In den selbst durchgeführten Usability-Tests konnten keine Usability-Probleme im Umgang mit der Darstellung der Awareness-Informationen beobachtet werden.

5.4 Auswertung der Fragebögen zu den Prototypen

Die im Anschluss an die Evaluation der Prototypen durch die Probanden anonym ausgefüllten Fragebögen, unterscheiden sich deutlich von den Antworten nach den Tests des unveränderten Saros-Release, die sich zum Vergleich im Anhang 7.4 finden.

Die Probanden bewerteten die Performance und die Komplexität des Einladungsprozesses zuvor negativer. Die Zustimmung zum Wunsch, dass Saros schneller arbeitet, verringerte sich.

1. Ich empfinde den Einladungsprozess von Saros als kompliziert
   - 25% Ich stimme eher zu
   - 50% Ich stimme eher nicht zu
   - 25% Ich stimme gar nicht zu
2. Der Einladungsprozess von Saros ist schnell
   - 100% Ich stimme eher zu
3. Ich wusste zu jedem Zeitpunkt ob / das Saros gerade aktiv ist und etwas tut
   - 75% Ich stimme eher zu
   - 25% Weder noch
4. Ich wusste zu jedem Zeitpunkt was genau (technisch) Saros gerade tut
   - 75% Ich stimme eher zu
   - 25% Weder noch
5. Ich finde es gut wenn Saros mir in Fortschittsbalken Details anzeigt, wie z.B. "Calculating Checksums for file: project/xyz.java"
   - 50% Ich stimme voll zu
   - 50% Ich stimme eher zu
6. Ich würde mir wünschen das Saros schneller arbeitet
   - 25% Ich stimme voll zu
   - 25% Ich stimme eher zu
   - 50% Weder noch

7. Ich wusste zu jedem Zeitpunkt was die anderen Sitzungsteilnehmer gerade tun
    - 50% Ich stimme eher zu
    - 25% Weder noch
    - 25% Ich stimme eher nicht zu
8. Saros macht es mir leicht zu erkennen, welche Dateien, Ordner und Projekte Bestandteil der Sitzung sind
    - 25% Ich stimme eher zu
    - 50% Weder noch
    - 25% Ich stimme eher nicht zu
9. Ich würde Saros anderen Entwicklern empfehlen
    - 100% Ich stimme eher zu

Die Nutzer bezeichnen den Einladungsprozess in der zweiten Befragung zwar als weniger kompliziert, allerdings kann dies daran liegen, dass die Nutzer mehr Erfahrung bei der Nutzung von Saros gesammelt haben, und sich damit insgesamt sicherer bei der Bedienung fühlen, da die Übersichtlichkeit des Einladungsprozesses (zumindest aus eigener, subjektiver Sicht) nicht signifikant verbessert werden konnte.

Auffällig ist bei den Antworten eine deutliche Verbesserung der Bewertung der Performance, auch wenn hierzu nur wenige Optimierungen durchgeführt werden konnten. Dies kann darin vermutet werden, dass die Nutzer nun nicht nur sofort Rückmeldungen über den Erfolg ihrer Interaktion mit Saros erhalten (beispielsweise durch sofortiges, sichtbares Öffnen des Einladungs-Wizards), sondern auch ausreichend präzise Rückmeldung über die zu erwartende Laufzeit des Einladungsprozesses erhalten. Ebenso kann die sich schnell verändernde Darstellung von Informationen, über die Aktivitäten im Einladungsprozess, die Arbeitsgeschwindigkeit subjektiv höher erscheinen lassen (Vgl. [Frederickson1993]).

Aufgrund der geringen Anzahl der befragten Nutzer sind diese Daten wenig aussagekräftig. Es ist möglich, dass die Rückmeldungen der Nutzer deutlich positiver ausgefallen sind, da diese in die Verbesserung der Usability involviert waren, was nach Nielsen (1993) Einfluss auf die subjektive Zufriedenheit der Nutzer nehmen kann.

Auch ist es möglich, dass die Nutzer sich über den Testverlauf so sehr in das System eingearbeitet haben, dass in späteren Evaluationen von Prototypen, Probleme nicht mehr beobachtet wurden, da die Probanden diesen (unbewusst) aus dem Weg gegangen sind.

Dennoch sehe ich diese Aussagen als ausreichend, um zu verdeutlichen, das eine positive Veränderung herbeigeführt wurde, insbesondere in Hinsicht auf die subjektive Zufriedenheit der Nutzer.

Nach der Betrachtung der Arbeitspakete, und den Ergebnissen der Evaluation der erstellten Prototypen, werden im folgenden, letzten Kapitel die Ergebnisse der Analyse und Überarbeitung der Usability von Saros zusammengefasst. Anschließend wird ein Ausblick gegeben, welche Probleme noch offen sind und wie diese behandelt werden können.

6 Fazit

In dieser Diplomarbeit wurden theorieorientierte, analytische und nutzerbasierte, empirische Evaluationsverfahren hinsichtlich ihrer Eignung für die Analyse der Usability von Saros untersucht.

Auf Basis der Erkenntnisse, die bei der Analyse von Nutzungsstatistik und Nutzer-Umfragen gewonnen wurde, wurden Testszenarien für die Durchführung der Usability-Tests konstruiert. Diese sollten sowohl die tatsächliche Nutzung von Saros berücksichtigen, als auch die Anwendungsszenarien, für die Saros geeignet sein soll. In einer Kombination aus heuristischer Analyse und Usability-Tests, unter Verwendung eines Codiscovery-Learning Ansatzes, wurden anschließend 128 Usability Probleme identifiziert, gewichtet und in einem Problemkatalog erfasst.

Für 54 Probleme wurden Lösungen konzipiert, die in einem iterativen Prozess, aus Entwicklungsarbeiten und weiteren Usability-Tests umgesetzt wurden. Der Fokus lag hierbei auf der Verbesserung des Systems, durch mehr Rückmeldungen über Systemaktivitäten, verstärkte Darstellung von Awarenessinformationen, sowie einer Überarbeitung des Einladungsprozesses.

Der Empfehlung von Nielsen (1993) folgend, wurden insgesamt drei Iterationen, aus Entwicklung und Evaluation der Änderungen, durchgeführt. Eine Verbesserung der Usability von Saros konnte dabei durch kontinuierliche Evaluation der Prototypen bestätigt werden. Beim Vergleich der Antworten der Probanden in standardisierten Fragebögen, vor und nach den Anpassungen an Saros, wurde eine deutliche, positive Veränderung in der subjektiven Zufriedenheit der Nutzer festgestellt.

In den Evaluationen konnte eine Verringerung der verbalen Koordination der Probanden untereinander beobachtet werden, auch wenn die Aussagekraft der Evaluationen, aufgrund der geringen Anzahl an Probanden, eingeschränkt ist. Die Durchführung weiterer Tests zur Evaluation der Usability von Saros ist somit notwendig, auch um die Out-of-Box-Experience zu überprüfen, die in dieser Arbeit nicht evaluiert wurde.

Bei der Wahl der Aufgaben für die Nutzertests wurden Aufgaben gewählt, die unter Einsatz von Saros gelöst werden können, ohne hierbei direkt auf die Funktionen von Saros einzugehen. Durch diesen Ansatz konnten Erkenntnisse über die Erlernbarkeit von Saros und die Gebrauchstauglichkeit der Funktionen gewonnen werden. Allerdings konnten dadurch einige Funktionen nicht auf Ihre Usability hin überprüft werden, da diese Funktionen durch die Nutzer nicht verwendet wurden. Offen bleibt, ob diese Funktionen sinnvoller Bestandteil von Saros sind, oder entfernt werden sollten.

Die Anzahl der gravierenden Usability-Probleme in Saros ist nach der Bearbeitung weiterhin hoch. Dies liegt vor allem an den experimentellen Funktionen, die in dieser Arbeit nicht bearbeitet wurden, aber einen Großteil der Usability-Katastrophen im Problemkatalog darstellen.

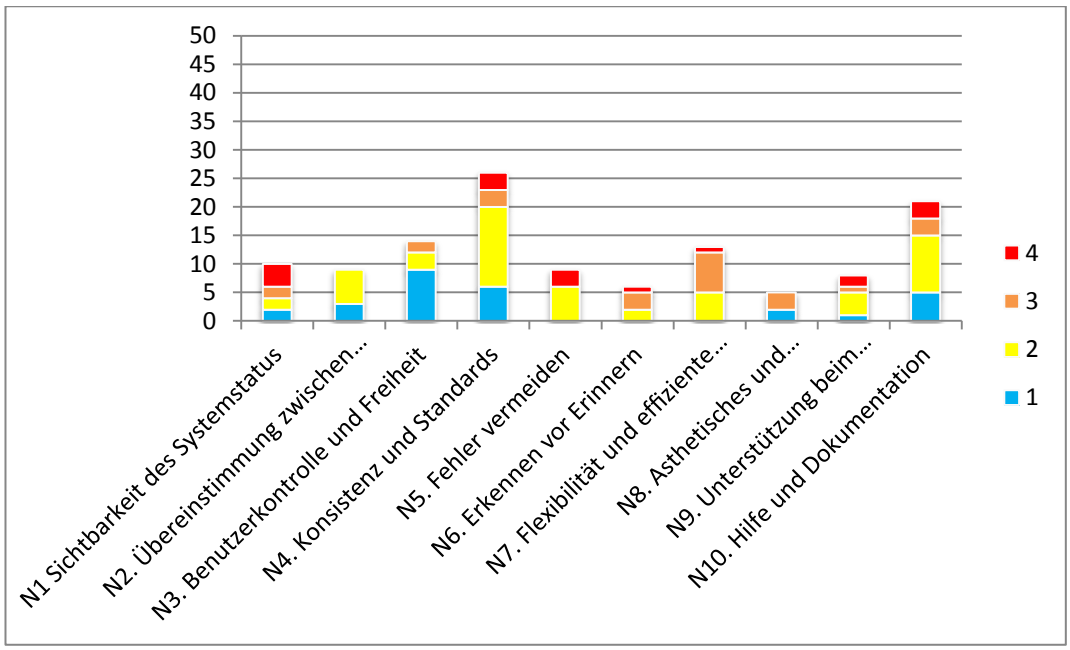

Abbildung 63: Verteilung der offenen Probleme nach Heuristiken

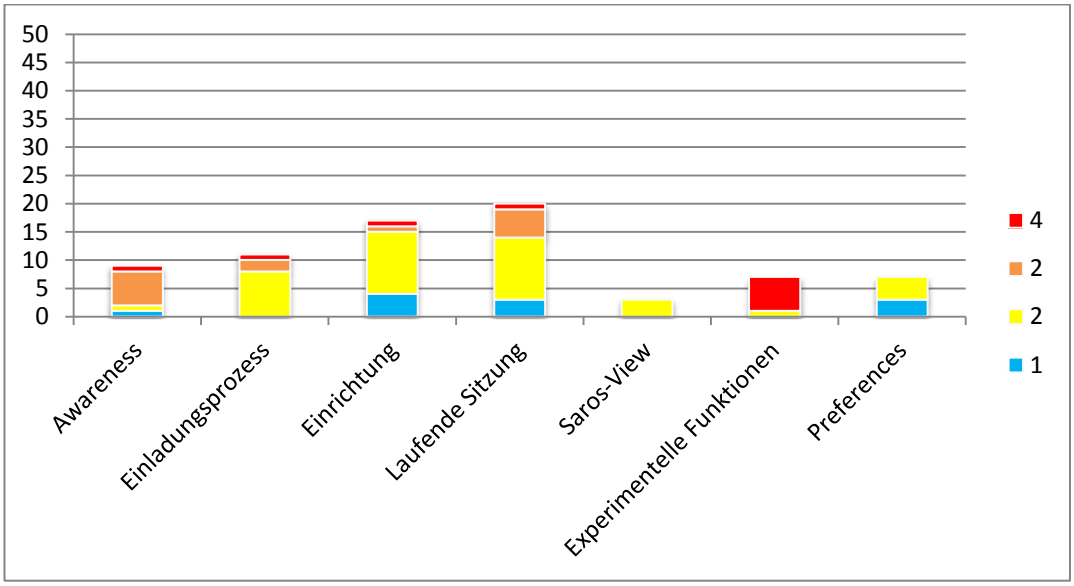

Abbildung 64: Verteilung der offenen Probleme nach Funktionsgruppen

Für die nicht bearbeiteten Probleme wurden erste Lösungsansätze ausgearbeitet, die auf den entwickelten Prototypen aufbauen. Da einige der Prototypen nicht in Nutzertests evaluiert werden konnten, verbleiben diese im Reviewboard des Saros-Teams, bis die Tauglichkeit durch eine Evaluation erfolgt ist.

Hierdurch wird vermieden, dass ungetestete Funktionen in Saros integriert werden, was ich als eine der Hauptursachen für die geringe Usability von Saros ansehe.

Die vollständige Liste der bearbeiteten und offenen Probleme findet sich im Problemkatalog im Anhang 7.7.

Einige der Probleme Probleme und Funktions-Wünsche der Nutzer, die in dieser Arbeit nicht behoben wurden, sollten hier erwähnt werden:

- Out-of-Box-Experience:
    - Die Einrichtung von Saros, insbesondere die Einstellungen sollten vereinfacht werden
    - Viele der Wizards, die bei der Einrichtung verwendet werden, sollten restrukturiert werden, da diese keine Wizards nach Eclipse Standard sind, und ein ungewöhnliches Layout aufweisen
- Tiefere Integration von Saros in Eclipse
    - Entwicklung einer eigenen Saros-Perspektive
    - Übertragung von Consolen-Ausgaben
    - Eigener Package Explorer mit besserer Visualisierung von Präsenzinformationen
    - Eigene Editor-Views
- Saros-Webseite
    - Außer der Seite „Usability Engineering“ und dem Tutorial für neue Saros-Nutzer wurde die Webseite nur an wenigen Stellen überarbeitet.
    - Ein kurzes Tutorial-Video, dass die Funktionen von Saros vorstellt
- Bessere Darstellung von Awareness-Informationen
    - In welcher Perspektive befindet sich der Nutzer
    - Welche Aktionen führt der Nutzer aus (z.B. Suche, Kompilieren, Applikation starten, Debuggen)
    - Telepointer oder andere bessere Indikationen für Klickaktionen innerhalb eines Editors
- Diverse Beschriftungen/Texte überarbeiten
    - „Buddies“, „XMPP/Jabber ID“, „Roster“, etc.
    - Dialogtexte
    - Einleitungstexte von Wizards / Wizard-Titel

Der letzte Punkt „Beschriftungen/Texte“ ist näher zu erläutern:

Grundsätzlich sollten überall einheitliche Bezeichnungen verwendet werden. Bei der Prüfung der Begriffe in Saros fiel so z.B. auf, dass Begriffe aus der Terminologie von XMPP übernommen wurden, um die Bezeichnungen in Saros zu vereinheitlichen. Nutzer werden als „Buddies“ hinzugefügt, was allerdings ein unangebrachter Begriff ist. Nicht jeder Arbeitskollege mit dem man über Saros zusammenarbeitet ist ein „Buddy“ was auf Deutsch Kumpel (oder Kamerad) bedeutet. Eine dahingehende Rückabwicklung der Änderung wurde im Bugtracker als Aufgabe für „später“ hinterlegt, da festgestellt werden musste, dass das Saros Test Framework (STF) diverse Bezeichnungen in Konfigurationsdateien redundant verwaltet, sodass hier eine Anpassung unverhältnismäßig aufwändig gewesen wäre. Zudem konnte in der Analyse kein in der Praxis auftretendes Usability-Problem entdeckt werden, möglicherweise deshalb, weil die Probanden sich tatsächlich als „Buddies“ ansehen.

Derartige Unstimmigkeiten existieren noch an diversen Stellen in Saros. So ist z.B. die Einrichtung von Saros nach wie vor von technischen Begriffen dominiert,

konnte aber im Rahmen dieser Arbeit nicht weiter bearbeitet werden. Eine entsprechende Restrukturierung der Wizards, und Überarbeitung der Texte bleibt somit offen.

6.1 Rückblick auf die Nutzungsstatistik

Nach Analyse der Ergebnisse und bei Betrachtung der Nutzungsstatistik kann nicht ausgeschlossen werden, dass Nutzer keine größeren Projekte mit Saros bearbeiten, weil sie Saros dafür als ungeeignet ansehen. So traten bei Tests der Prototypen mit dem Saros-Team etliche Probleme mit dem Arbeitsspeicherverbrauch auf, bereits bei Projekten mit einer Größe von 100 MB auf Laptops mit 2GB Arbeitsspeicher, die zu Abstürzen von Eclipse führten. Durch diese mangelnde Stabilität besteht die Möglichkeit, dass Nutzer auf Computern mit wenig Arbeitsspeicher oder ungünstigen Systemeinstellungen (z.B. abgeschalteter Auslagerungsdatei) gar nicht in der Lage sind, größere Projekte mit Saros zu bearbeiten.

Dies hat insbesondere Auswirkungen auf die Frage nach dem Bedarf einer Funktionalität für Sitzungen mit mehr als 2 Teilnehmern. Die Nutzungsstatistik zeigt deutlich, dass die Saros-Nutzer fast ausschließlich zu zweit gearbeitet haben. Dies kann allerdings darin begründet sein, dass weder die Stabilität noch die Geschwindigkeit, mit der mit Saros gearbeitet werden kann, die Nutzer in Sitzungen mit 2 Personen zufrieden stellt, und sie daher Sitzungen mit mehr Teilnehmer gar nicht erst ausprobieren.

Auch die extrem geringe Nutzungsfrequenz der VoIP-Funktionalität kann viele Gründe haben:

- Nutzer finden die Funktion nicht (nur im Kontextmenü sichtbar)
- Nutzer scheitern an der Einrichtung
- Nutzer testen die Funktionalitäten einmalig, sind aber unzufrieden
    - Deutliche Verzögerungen in der Übertragung (Latenz)
    - Es fehlt z.B. die Rauschunterdrückung (man hört Tastaturgeklapper sehr deutlich)
- Nutzer sind schon per Skype miteinander verbunden und haben keinen Bedarf

So bleibt es offen zu entscheiden, ob es tatsächlich Sinn macht, z.B. die VoIP-Komponente weiter zu entwickeln, sodass diese einsatzfähig wird. Priorität sollte die Screensharing-Komponente bekommen, die eine definitiv nützliche Funktion ist, allerdings in ihrer jetzigen Umsetzung ungeeignet für den praktischen Einsatz.

6.2 Vorschläge zur Verbesserung der Prozesse

Der bisherige Entwicklungsprozess im Saros-Team ist vor allem auf technische Qualität der Umsetzung neuer Funktionen oder Veränderungen bestehender Funktionen fokussiert. Eine formale Integration des Usability Engineering in den Entwicklungsprozess erscheint mir sinnvoll, wobei dies eher als eine Hilfestellung für Entwickler konstruiert werden sollte.

Die Usability-Engineering Unterseite von der Saros-Webseite wurde bereits überarbeitet. Neben dem Entfernen diverser falscher Angaben, wurden die 10

Heuristiken von Jakob Nielsen integriert, um eine schnelle Übersicht der zu befolgenden Gestaltungsgrundsätze für die Entwickler bereitzustellen. Grundsätzlich sollte die Berücksichtigung der Usability-Richtlinien eine Grundvoraussetzung für weitere Veränderungen der Saros-Oberfläche sein.

Neben der Qualitätskontrolle des Codes über das Reviewboard, könnte ein Prozess etabliert werden, der regelmäßige Nutzertests des Systems vorsieht, und die einfache Erfassung von identifizierten Usability-Problemen ermöglicht.

Der Bugtracker auf Sourceforge erscheint für die Ablage der Usability-Probleme weniger geeignet, da hier Usability-Probleme mit technischen Defekten gemischt werden. Durch den Aufbau eines separaten Trackers für die Usability-Probleme wäre es zwar schwieriger, eine einheitliche Priorisierung der Probleme in Saros vorzunehmen, allerdings zeigte sich, dass einige Entwickler sich ausschließlich der technischen Probleme annehmen, und wenig an Usability-Probleme interessiert sind. Da die Filterung des Bugtrackers bei Sourceforge stark ausbaufähig ist, ist es diesen Entwicklern nicht möglich, Usability-Probleme auszublenden.

Es bleibt somit offen, wie künftig Usability-Probleme erfasst werden können, und sichergestellt wird, dass diese auch mit der nötigen Priorität behandelt werden. Ein erster Ansatz durch den Aufbau einer relationalen MySQL-Datenbank existiert durch diese Diplomarbeit, auch wenn noch verständliche Vorbehalte, gegen eine separate Datenhaltung, im Saros-Team existieren.

Weitere Abschlussarbeiten zum Thema Usability sind geplant und zum Teil bereits in Arbeit, sodass es wahrscheinlich ist, dass die Usability von Saros sich in Zukunft deutlich verbessern wird.

Als eine mögliche Prozessveränderung, zur Verbesserung der Datenbasis für diese Arbeiten, könnten die Nutzer bei Deaktivierung der Nutzungsstatistik gefragt werden, wieso sie diese deaktivieren, um hier bessere Einblicke zu erhalten.

Auch sollte die Nutzungsstatistik selbst deutlich ausgeweitet, und professionalisiert werden. Es fehlten insbesondere Daten über

- Tatsächliche Projektgröße (getrennt von Datenmenge insgesamt)
- Häufigkeit des nachträglichen Einladens von Nutzern zu einer laufenden Sitzung
- Konfiguration von Saros (Screensharing, sonstige Preferences)
- Dauer der Nutzerinteraktion z.B. mit den neuen Wizards

Zudem wäre es sinnvoll, wenn die Statistik mehr Möglichkeiten zur Auswertung bereitstellt, und vor allem automatisch den Datenmüll entfernt (z.B. DEVEL-Versionen von Entwicklern), der die Statistik verfälscht ohne Mehrwert zu bieten.

Eine weitere Überarbeitung der Webseite ist dringend notwendig. So sind diverse veraltete Dokumentationen der Systemarchitektur in der Seite verstreut. Auch die Erstellung eines neuen Saros-Tutorial-Videos wäre sinnvoll, das neben einer Anleitung zur Einrichtung des Systems auch Einblicke in den Funktionsumfang von Saros gibt.

Um die Datenbasis für zukünftige Usability-Analysen und Verbesserungen zu vergrößern, sollten die Akzeptanz-Tests vor einem Release per Video aufgezeichnet werden. Dies würde es ermöglichen, Fehler leichter nachzuvollziehen (Nutzer erinnern sich erfahrungsgemäß schlecht daran, was genau sie gemacht hatten bevor ein Fehler auftrat) und eine Analyse der Videoaufzeichnungen könnte Hinweise auf Usability-Probleme liefern.

Die Saros-Nutzerumfrage sollte standardisiert werden und mehr Fragen umfassen, darunter die Abfrage der subjektiven Bewertungen der Geschwindigkeit und der Komplexität des Systems auf einer Likert-Skala, um mehr Informationen über die subjektive Zufriedenheit der Nutzer zu erhalten.

7 Anhang

In diesem Anhang findet sich das, was in dieser Arbeit nicht mehr erwähnt werden konnte, ohne dadurch den Lesefluss zu stören.

Wichtigster Bestandteil dieses Anhangs ist der Problemkatalog in Abschnitt 7.7, in dem alle identifizierten Probleme

7.1 Übersicht der Usability Methoden von Nielsen

Method Name	*Lifecycle Stage*	*Users Needed*	*Main Advantage*	*Main Disadvantage*
Heuristic evaluation	Early design, "inner cycle" of iterative design	None	Finds individual usability problems. Can address expert user issues.	Does not involve real users, so does not find "surprises" relating to their needs.
Performance measures	Competitive analysis, final testing	At least 10	Hard numbers. Results easy to compare.	Does not find individual usability problems.
Thinking aloud	Iterative design, formative evaluation	3–5	Pinpoints user misconceptions. Cheap test.	Unnatural for users. Hard for expert users to verbalize.
Observation	Task analysis, follow-up studies	3 or more	Ecological validity; reveals users' real tasks. Suggests functions and features.	Appointments hard to set up. No experimenter control.
Questionnaires	Task analysis, follow-up studies	At least 30	Finds subjective user preferences. Easy to repeat.	Pilot work needed (to prevent misunderstandings).
Interviews	Task analysis	5	Flexible, in-depth attitude and experience probing.	Time consuming. Hard to analyze and compare.
Focus groups	Task analysis, user involvement	6–9 per group	Spontaneous reactions and group dynamics.	Hard to analyze. Low validity
Logging actual use	Final testing, follow-up studies	At least 20	Finds highly used (or unused) features. Can run continuously.	Analysis programs needed for huge mass of data. Violation of users' privacy.
User feedback	Follow-up studies	Hundreds	Tracks changes in user requirements and views.	Special organization needed to handle replies.

Abbildung 65: Übersicht der Usability Verfahren mit Vorteilen und nötiger Probanden-Anzahl [Nielsen1993, S.224, Table 10 Summary of the usability methods]

7.2 Die12 Heuristiken von Sarodnick und Brau

1. Aufgabenangemessenheit

Alle benötigten Funktionen für anstehende Aufgaben im System müssen vorhanden und hinreichend so gestaltet sein, dass sie den Nutzer unterstützen und bei Routineaufgaben entlasten.

2. Prozessangemessenheit

Das System sollte für die Erfüllung realer Arbeitsaufgaben in typischen Einsatzfeldern optimiert sein, einen Bezug zum übergeordneten realen Prozessziel haben und auf Qualifikationen und Erfahrungen der realen Nutzer abgestimmt sein.

3. Selbstbeschreibungsfähigkeit

Einheitliche und unmittelbare Anzeige des Systemstatus. Der Benutzer sollte die Detaillierung der Information über den Systemstatus bestimmen können

4. Steuerbarkeit

Beinhaltet die Kontrolle des Nutzers über den Dialog, sowie die Möglichkeit, verschiedene Eingabehilfen zu nutzen oder das System ohne Datenverlust zu beenden.

5. Erwartungskonformität

Die Informationsdarstellung sollte systemimmanent und mit plattformspezifischen Konzepten konsistent sein. Bei ähnlichen Aufgaben sollten Dialoge vergleichbar und an erwarteter Position dargestellt sein.

6. Fehlertoleranz

Fehlermeldungen sollten deutlich sein und Hinweise beispielsweise über Art und Handlungszusammenhang enthalten. Der Nutzer muss über irreversible Handlungen informiert werden.

7. System- und Datensicherheit

Das System sollte auch bei fehlerhaften Eingaben des Nutzers und unter hoher Ressourcenbelastung stabil und ohne Datenverluste arbeiten.

8. Individualisierbarkeit

Das Dialogsystem sollte sich individuell an die Präferenzen der Nutzer anpassen lassen, solange dies der Effektivität, Effizienz und Zufriedenstellung dient und nicht im Widerspruch zu notwendigen technischen oder sicherheitsrelevanten Begrenzungen steht.

9. Lernförderlichkeit

Lernstrategien wie "Learning by Doing" sollten durch schrittweise Anleitungen oder Navigationshilfen unterstützt werden.

10. Wahrnehmungssteuerung

Das Layout sollte minimalistisch gehalten werden. Gruppierungen, Farbgestaltung und sinnvolle Informationsreduktion etc. sollten so verwendet werden, dass die Aufmerksamkeit des Nutzers hin zu relevanter Information gelenkt wird.

11. Joy of use

Arbeitsabläufe und grafische Gestaltung des Systems sollten bei notwendiger Konsistenz Monotonie vermeiden und zeitgemäß wirken. Metaphern sollten adäquat und auf den Nutzungskontext abgestimmt verwendet werden.

12. Interkulturelle Aspekte

Das System sollte auf einen definierten Nutzerkreis und dessen funktionale, organisatorische und nationale Kultur abgestimmt sein.

[Sarodnick2006]

7.3 Fortschrittsanzeigen in Eclipse

Bei den Arbeiten an der Verbesserung des Feedbacks über den Systemzustand wurde intensiv mit der Eclipse Jobs-API gearbeitet und geprüft, welche Möglichkeiten bereitstehen, um Fortschritte anzuzeigen. Leider sind viele der Funktionen die Eclipse bereitstellt, nur bedingt einsetzbar oder funktionierten aus Gründen nicht, die bis jetzt nicht identifiziert werden konnten.

Eclipse bietet mehrere Möglichkeiten, Fortschrittsanzeigen einzublenden.

Die einfachste wäre der *BusyIndicator* eine Klasse, die den Mauszeiger des Nutzers festsetzt auf den „Blockierende Operation" Mauszeiger – unter Windows ist dies z.B die bekannte Sanduhr, solange das übergebene *Runnable*-Objekt aktiv ist.

```
org.eclipse.swt.custom.BusyIndicator

public static void showWhile(Display display, Runnable runnable)
```

Dies ist meiner Ansicht nach nur sinnvoll für Prozesse, die wenige Sekunden dauern oder sonst z.B. zu einem Flackern des Mauszeigers führen. Da keine Information gegeben wird, was passiert und wie lange dies noch dauern kann, ist dies eher ungeeignet für Prozesse die länger als 3-5 Sekunden laufen (Vgl. [Nielsen1993])[26].

Für die Darstellung von einfachen Fortschrittsbalken bietet Eclipse den *ProgressMonitorDialog* an:

```
ProgressMonitorDialog dialog = new ProgressMonitorDialog(shell);

dialog.run(true, true, new IRunnableWithProgress(){

   public void run(IProgressMonitor monitor) {

      monitor.beginTask("Doing work here ...", 100);

      // do some work…

      monitor.worked(1); // Report Progress in "ticks"

      // do more work..

      monitor.done();  // finished the job…

   }

});
```

Diese Art von Dialog wird modal angezeigt, d.h. blockierend. Somit ist es für den Nutzer nicht möglich, im Hintergrund in Eclipse zu arbeiten, während der Prozess läuft. Dazu müsste der Prozess zunächst abgebrochen werden.

Dies ist nur für Prozesse sinnvoll, die tatsächlich den Nutzer blockieren müssen, da nebenläufige Veränderungen z.B: im Workspace zu Problemen führen würden.

[26] Auch lesenswert: http://www.useit.com/papers/responsetime.html

Für alle anderen Prozesse, die sowohl abbrechbar sein sollen, als auch im Hintergrund laufen können, bietet sich sich die Eclipse Jobs-API an.

```
Job job = new Job("My new job") {
   @Override
   protected IStatus run(IProgressMonitor monitor) {
      monitor.beginTask("Doing work here ...", 100);
      // do some work…
      monitor.worked(1); // Report Progress in "ticks"
      // do more work..
      monitor.done();  // finished the job…
      return Status.OK_STATUS;
   }
};
job.setUser(true)
job.schedule();
```

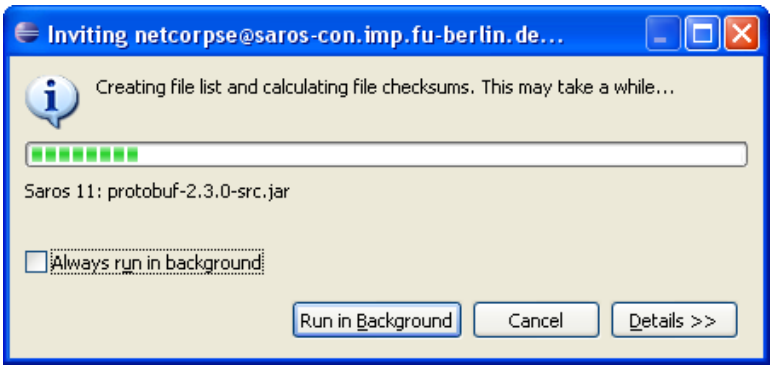

Abbildung 66: Fortschrittsanzeige eines User-Jobs

Hierbei gibt es mehrere Arten von Jobs:

- User-Job: ein Dialog mit Fortschrittsbalken wird angezeigt (siehe Screenshot)
    - Dialog kann in den Hintergrund geschickt werden mit „Run in Background"
    - Checkbox „Always run in background" automatisiert dies für den Nutzer bei zukünftigen identischen Jobs
- Background-Jobs (es wird kein Dialog geöffnet)
    - Erscheinen im Eclipse-Progress-View
- System-Jobs
    - laufen unsichtbar im Hintergrund, ohne dass der Nutzer etwas davon sieht

Jobs können unterschiedliche Prioritäten haben, und werden entsprechend der Priorität von Eclipse ausgeführt. Hierbei wird zwischen *Job.SHORT* und *Job.LONG* unterschieden.

Der Scheduler von Eclipse führt z.B. Jobs mit *Job.SHORT* vor den Jobs mit Priorität *Job.LONG* aus.

In langlaufenden Routinen kann in der *run()*-Methode eines Jobs mittels *monitor.isCanceled()* geprüft werden, ob der Nutzer den Prozess abbrechen möchte (die „Cancel"-Schaltfläche wurde betätigt). Je öfter dies geprüft wird, desto schneller bricht der Prozess für den Nutzer tatsächlich ab.

So konnte beispielsweise in Saros die Berechnung der Checksummen zwar abgebrochen werden über die „Cancel"-Schaltfläche im Fortschrittsdialog, allerdings wurde die Operation dennoch bis zu Ende durchgeführt. Dies führte zu enormer Frustration bei den Nutzern, da diese nun warten mussten, und keine Möglichkeit hatten, den Prozess tatsächlich zu unterbrechen.

Jobs können auf zweierlei Art und Weise gruppiert werden:

Durch implementieren der Methode

```
public boolean belongsTo(Object family)
```

können Jobs gruppiert werden, sodass dann über einen *IJobManager* alle Jobs einer Familie pausiert oder sogar abgeberochen werden können.

Hierzu wird über den Platform-JobManager und einen entsprechenden Schlüssel der gleichartige Jobs kennzeichnet (z.B. einen String) die Jobfamilie angehalten, oder komplett gestoppt.

```
// Obtain the Platform job manager

IJobManager manager = Platform.getJobManager();

// put the family to sleep

manager.sleep("Jones");

// put the family to sleep for good!

manager.cancel("Jones");
```

In den Jobs ist es auch möglich, über die Methode *schedule(long time)* einen Job erneut ausführen zu lassen zu einem späteren Zeitpunkt (z.B. für regelmäßige Aktivitäten, wie das Prüfen der Konsistenz durch den Watchdog in Saros).

Wer das Scheduling der Jobs noch weiter treiben möchte, kann sich mit der *ISchedulingRule* beschäftigen, die zudem noch mit *MultiRule* kombiniert werden kann, um so komplexe Abhängigkeiten von Jobs behandeln zu können.

Die zweite Möglichkeit, Jobs zu gruppieren wäre über den Einsatz einer *ProgressGroup*. Hierbei wird ein langlaufender Prozess in Teilprozesse zerlegt, die jeweils als eigener Job ausgeführt werden (z.B. parallel laufend) und in einem übergeordneten Fortschrittsbalken dargestellt werden.

Es wurde zwar versucht, dies für die Saros-Jobs zu verwenden, die in einem Einladungsprozess ablaufen, um diese zusammen zu fassen, und für den Nutzer somit einheitlicher zu wirken. Allerdings ist hierbei das Problem, dass sich aus unerfindlichen Gründen einzelne Jobs nicht mehr abbrechen ließen, möglicherweise aufgrund eines Bugs innerhalb von Saros, der leider nicht gefunden wurde.

Somit müsste der einladende Nutzer die Einladung für alle Nutzer zusammen abbrechen, da es nicht möglich war, eine einzelne Einladung abzubrechen, wenn diese zusammengefasst wurde in einer ProgressGroup.

```
Job jobOne = new Job(„Erster Job“);

Job jobTwo = new Job(“Zweiter Job”);

IProgressMonitor group = Platform.getJobManager().createProgressGroup();

Group.beginTask(“Working in two jobs parallel”,100);

jobOne.setProgressGroup(group,50);

jobOne.schedule();

jobTwo.setProgressGroup(group,50);

jobTwo.schedule()
```

Code X: Gruppieren von Jobs die parallel laufen

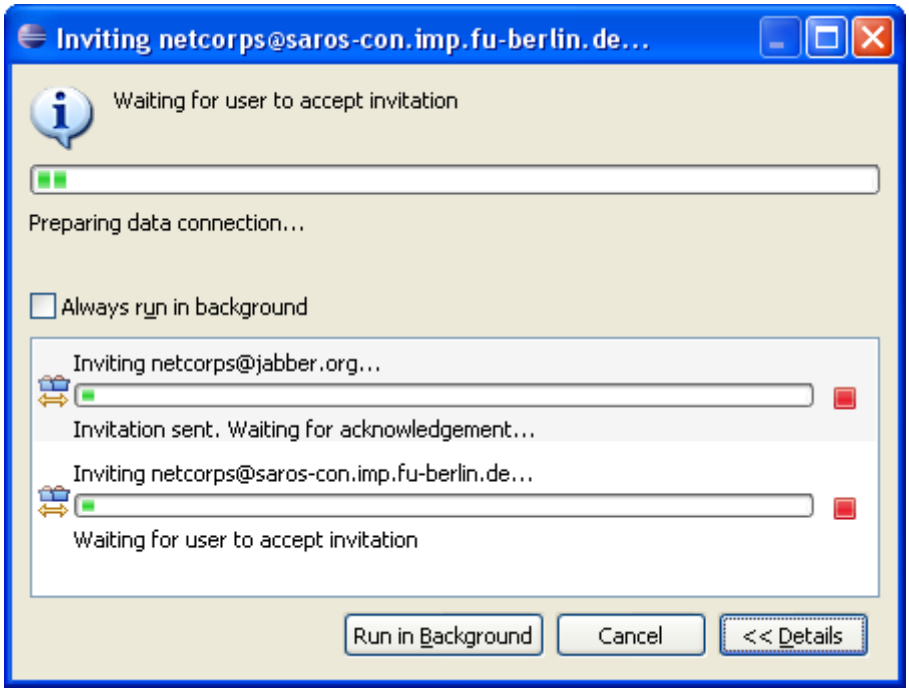

Abbildung 67: Zwei zusammengefasste Einladungsprozesse

Schickt der Anwender den Fortschrittsdialog in den Hintergrund so ist dieser danach im Eclipse Progress View sichtbar:

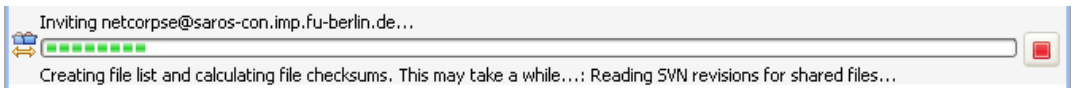

Abbildung 68: Eclipse Progress View

Nähere Informationen darüber, wie Jobs eingesetzt werden sollten, und welche Methoden noch zur Verfügung stehen findet sich auf der Eclipse-Webseite[27].

Weitere interessante Tips finden sich auf im Eclipse-Blog[28], wobei hier sehr detailliert auf andere Möglichkeiten, Fortschritte anzuzeigen eingegangen wird, unter anderem in dem z.B. der Titel eines Views verändert wird oder in der Statusleiste von Eclipse ein Miniatur-Fortschrittsbalken eingeblendet wird.

Die Interessanteste Alternative zu den Job-Fortschrittsdialogen sind Fortschrittsbalken innerhalb eines Wizards. In Wizard Dialogen kann ein integrierter Fortschrittsbalken verwendet werden, der im unteren Teil des Wizards angezeigt wird

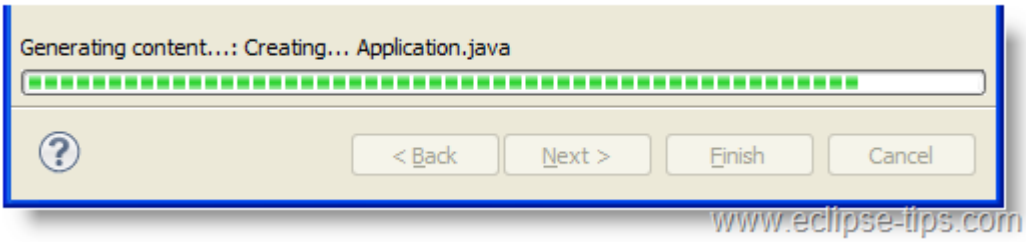

Abbildung 69: Fortschrittsbalken in einem Wizard (Bildquelle: www.eclipse-tips.com)

Auf den Einsatz dieser Technik wurde allerdings in Saros verzichtet, da der Einsatz dieser Fortschrittsbalken bedeutet, dass der Wizard weiterhin modal geöffnet bleibt, und somit einen Großteil der Bildschirmfläche blockiert. Solange der dem Wizard übergebene Prozess läuft, sind alle Interaktionselemente des Wizards deaktiviert.

Denkbare Einsatzfälle wären hier langlaufende Aktivitäten, die von einem Schritt eines Wizards zum nächsten Schritt des Wizards ausgeführt werden müssen. In Saros gab es somit keinen Anwendungsfall für diese Art der Fortschrittsanzeige.

---

[27] http://www.eclipse.org/articles/Article-Concurrency/jobs-api.html
[28] http://blog.eclipse-tips.com/2009/02/using-progress-bars.html

7.4 Auswertung der Fragebögen

Die Probanden wurden nach den Usability-Tests der unveränderten Saros-Release-Version gebeten, einen standardisierten Fragebogen auszufüllen, mit folgenden Ergebnissen:

1. Ich empfinde den Einladungsprozess von Saros als kompliziert
    - 75% Ich stimme eher zu
    - 25% Ich stimme eher nicht zu
2. Der Einladungsprozess von Saros ist schnell
    - 25% Weder noch
    - 50% Ich stimme eher nicht zu
    - 25% Ich stimme gar nicht zu
3. Ich wusste zu jedem Zeitpunkt ob / das Saros gerade aktiv ist und etwas tut
    - 50% Ich stimme eher nicht zu
    - 50% Ich stimme gar nicht zu
4. Ich wusste zu jedem Zeitpunkt was genau (technisch) Saros gerade tut
    - 50% Ich stimme eher nicht zu
    - 50% Ich stimme gar nicht zu
5. Ich finde es gut wenn Saros mir in Fortschrittsbalken Details anzeigt, wie z.B. "Calculating Checksums for file: project/xyz.java"
    - 50% Ich stimme voll zu
    - 50% Ich stimme eher zu
6. Ich würde mir wünschen das Saros schneller arbeitet
    - 25% Ich stimme voll zu
    - 50% Ich stimme eher zu
    - 25% Weder noch
7. Ich wusste zu jedem Zeitpunkt was die anderen Sitzungsteilnehmer gerade tun
    - 75% Ich stimme eher nicht zu
    - 25% Ich stimme gar nicht zu
8. Saros macht es mir leicht zu erkennen, welche Dateien, Ordner und Projekte Bestandteil der Sitzung sind
    - 50% Ich stimme eher nicht zu
    - 50% Ich stimme gar nicht zu
9. Ich würde Saros anderen Entwicklern empfehlen
    - 50% Ich stimme eher zu
    - 25% Weder noch
    - 25% Ich stimme eher nicht zu

7.5 Kontextmenü-Strukturen nach Microsoft Guideline

- Organize the menu items into groups of seven or fewer strongly related items. For this, submenus count as a single menu item in the parent menu.
- Don't put more than 25 items within a single level of a menu (not counting submenus).
- Put separators between the groups within a menu. A separator is a single line that spans the width of the menu.

- Within a menu, put the groups in their logical order. If there is no logical order, place the most commonly used groups first.
- Within a group, put the items in their logical order. If there is no logical order, place the most commonly used items first. Put numeric items (such as zoom percentages) in numeric order.

Quelle: [MicrosoftGuideline]

7.6 Die Arbeit im Saros-Team

Da Saros parallel von anderen Entwicklern bearbeitet wurde, war es erforderlich, die Arbeiten mit diesen zu koordinieren. Die Vorgehensweise bei der Koordination mit dem Saros-Team und die Prinzipien denen bei der Entwicklung gefolgt wurde, sind hier erwähnt.

7.6.1 Koordination der Entwicklungsarbeiten

Die Software-Fehler die während der Arbeit an Saros, und insbesondere bei der Durchführung der heuristischen Analyse identifiziert wurden, wurden umgehend im SourceForge Bugtracker gemeldet, damit andere Saros-Entwickler von deren Existenz Kenntnis nehmen können.

Bei Problemen, die im direkten Zusammenhang mit den durchzuführenden Anpassungen standen, habe ich den jeweiligen Bug explizit mir selbst zugewiesen um zu verdeutlichen, dass ich diesen Bearbeiten werde, und habe dies auch entsprechend in der Beschreibung des Bugs erwähnt.

Insgesamt wurden über 40 Bugs, darunter mehrere schwerwiegende Fehler erfasst. Zwei der größten Probleme die bei Tests identifiziert wurden, beschäftigten einen Teamkollegen im Rahmen seiner Masterarbeit nahezu 2 Monate, da hierbei festgestellt wurde, dass die Komponente mit dem Fehler zu großen Teilen umgeschrieben werden muss.

Neben den wöchentlichen Meetings im Saros-Raum am Fachbereich Informatik gab es regen Austausch über die Saros-Developer-Mailingliste. Zudem erfolgte ein intensiver Austausch mit den Teamkollegen via Skype-Chat und Skype-Telefonie.

7.6.2 Qualitätssicherung

Änderungen am System mussten grundsätzlich als möglichst kleiner SVN Patch im teaminternen Review-Board hochgeladen werden und durften nicht direkt in das SVN Repository committed werden.

Durch das Bereitstellen der Patches im Reviewboard wurde sichergestellt, dass die anderen Saros-Entwickler mitbekommen, welche Änderungen am System erfolgen, und sie hatten die Gelegenheit die Code-Qualität zu prüfen und mögliche Bedenken zu äußern. Bevor eine Änderung veröffentlicht werden durfte, mussten zwei Entwickler ein Review durchgeführt haben, und über das System ihr „OK“ geben. Durch diese 6 Augen Prinzip konnte während der Bearbeitungszeit dieser Diplomarbeit sichergestellt werden, dass (fast) kein defekter Code committed wurde. Auch gab es hier Feedback zur

Oberflächengestaltung und möglichen Randbedingungen, die bei der Entwicklung zuvor übersehen wurden.

Das Problem mit dem Reviewboard bestand vor allem darin, dass vor Bereitstellung eines Patches, die Prototypen für die Evaluierungen entwickelt wurden und sich zunächst in einem Usability-Test beweisen mussten. Da es nicht möglich wäre, jede kleine Veränderung einzeln zu testen mit den Probanden, mussten so mehrere Anpassungen auf einmal getestet werden. Entsprechend aufwändig war danach die Trennung der einzelnen Änderungen in Patches. Es hatte sich allerdings in der Vergangenheit gezeigt, dass große Patches, für deren Review länger als 15 Minuten gebraucht wird, selten reviewed werden und es somit unverzichtbar ist, die durchgeführten Entwicklungsarbeiten sorgsam zu separieren.

Um sicherzustellen, dass die neuen Funktionen und geänderten Komponenten auch nach Anpassungen anderer Entwickler in Zukunft verlässlich funktionieren, wurden automatisierte Software-Tests geschrieben. Dies ermöglicht es anderen Entwicklern, das fehlerfreie Funktionieren der Komponenten nach eigenen Anpassungen des Codes zu verifizieren.

Hierbei kamen Tests über das Saros Test Framework (STF) zum Einsatz. Das STF ist durch eine Bot-Implementierung in der Lage, die Oberfläche von Saros anzusteuern, und dort mit Bedienelementen zu interagieren. Es ermöglicht somit, so gut wie alle Oberflächenkomponenten wie ein echter Nutzer zu bedienen, und ist auch in der Lage Interaktionen mehrerer Nutzer innerhalb einer Sitzung zu simulieren.

Die Fähigkeiten des Bots sind allerdings noch ausbaufähig, weshalb ein Großteil der Änderungen nicht automatisiert getestet werden kann. So ist es nicht möglich die Anzeigen in Fortschrittsbalken zu testen, da hierzu sehr große Dateien generiert werden müssten, um eine ausreichend lange „Übertragungsdauer“ zu erreichen und etwas aus den Fortschrittsbalken anzeigen lassen zu können. Die Tests wären somit in großem Umfang von der eingesetzten Hardware abhängig und könnten je nach System- und Netzwerklast sporadisch fehlschlagen.

Somit lag der Fokus bei der Entwicklung auf den stabil laufenden Tests, für die neuen Funktionen, wie beispielsweise Undo/Redo und dem Speichern von Selektionen für einen schnelleren Sitzungsstart.

Hierbei ist anzumerken, dass aufgrund der Komplexität des STF, Tests erst dann geschrieben wurden, wenn die entwickelten Prototypen sich im Praxiseinsatz bewähren konnten. So wurde verhindert, dass Zeit für das Schreiben automatisierter Tests von Komponenten aufgewandt wird, die durch Probleme die Evaluationen auftreten, verworfen werden müssen.

7.6.3 Entwicklungsprozess

Bei der Entwicklung wurden die grundsätzlichen Prinzipien der Objektorientierten Programmierung angewandt. Wo passende Entwurfsmuster einsetzbar waren, wurden diese verwendet, ohne dabei voreilig Optimierungen oder Generalisierungen durchzuführen.

Ein Problem der Codebasis von Saros ist unter anderem, dass viele Studenten, z.B. aus dem „Softwaretechnik Projekt“, meist ohne praktische Erfahrung in der Objektorientierten Programmierung, viele Klassen extrahiert haben, ohne dass dies Vorteile bietet. Auch wurde, wenn es Sinn machen würde eine Klasse zu extrahieren, stattdessen der Code mehrfach kopiert, sodass Änderungen z.B. am Einladungsprozess sich teilweise schwierig gestalteten, da hier ein größeres Refactoring notwendig war.

Bei der Entwicklung wurde daher dem Grundsatz „you touch it, you fix it“ gefolgt, was bedeutet, dass Klassen, die verändert oder genutzt wurden, angepasst werden, wenn diese deutliche Abweichungen zu den Saros-Codestandards aufweisen.

Leider ist es zeitlich nicht möglich gewesen, alle Schwachstellen im Code von Saros zu beseitigen. Damit das erlangte Wissen über ungünstige Implementierungen, die überarbeitet werden sollten, nicht verloren geht, wurde ein neuer Saros-interner Bugtracker eingerichtet, für „Refactoring Requests“. In diesem Tracker wird erfasst, wenn im Saros-Code konzeptuelle oder strukturelle Mängel vorliegen, die von demjenigen der sie entdeckt hat, nicht behoben werden konnten.

Die Bugtracker für funktionale/technische Fehler und der Dokumentations-Tracker wären für eine derartige Ablage ungeeignet gewesen, und der Tracker wurde binnen weniger Tage bereits umfangreich gefüllt.

7.6.4 Einarbeitung in den Saros-Code

Der Code von Saros ist aufgrund der Entwicklungsgeschichte, stellenweise schwer zu verstehen. Durch die Struktur von Saros durchzusteigen war eine Herausforderung, aber eine durchaus interessante. Neben dem Refactoring von allzu unsinnigem Code, wurden diverse JavaDoc-Kommentare für Methoden und Klassen angepasst, um es nachfolgenden Entwicklern zu erleichtern, sich in den Code einzuarbeiten.

„Kommentare schreiben um Code zu verstehen“ war gerade bei Saros eine effiziente Methode, um die Zusammenhänge der einzelnen Komponenten zu verstehen.

Auch für neu entwickelte Klassen wurden entsprechende JavaDocs geschrieben, wobei hier den Oracle Javadoc Guidelines[29] gefolgt wurde, die jeder Java-Entwickler (auch für andere Programmiersprachen sinnvoll) kennen sollte.

Methoden-Dokumentationen beschreiben vor allem wofür die Methode gedacht ist, wie man sie verwenden sollte und deren Nebeneffekte. Was nicht beschrieben wird, ist wie genau die Methode implementiert ist. Insbesondere wurden sinnlose Kommentare wie das klassische Anti-Beispiel von Oracle vermieden.

---

[29] http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

```
/**
 * Sets the tool tip text.
 *
 * @param text  the text of the tool tip
 */
public void setToolTipText(String text);
```

Derartige Kommentare helfen keinem Entwickler weiter, da sie nur die Methodennamen und Variablennamen wiederholen. Von dieser Art von Kommentar gibt es in Saros etliche – wo es ging wurden diese ersetzt oder entfernt, allerdings ist bei der Menge an Code eine komplette Überarbeitung nicht möglich gewesen.

7.7 Der Usability-Problemkatalog

In diesem Abschnitt findet sich der Katalog aller identifizierten Usability-Probleme. Ich erhebe nicht den Anspruch der Vollständigkeit für diesen Katalog. Es ist vielmehr davon auszugehen, dass noch weitere Usability-Probleme in Saros existieren, die ich nicht entdeckt habe.

Eine genaue Beschreibung des Aufbaus dieses Problemkatalogs findet sich in Abschnitt 3.9. Eine leichter zu durchsuchende und in naher Zukunft interaktiv zu filternde Ansicht findet sich im Internet [30]

Trigger: Probleme die in der Heuristischen Evaluation ermittelt wurden, sind mit „HE“ markiert. „UT“ weist auf Bestätigung/Erkenntnisse aus eigenen Usability-Tests hin. UT[QUELLE] weist auf die Auswertung der Videos der Usability Tests von Arsenij Solovjev oder auf Einträgen aus der Liste verbleibender Probleme aus der Arbeit von Björn Kahlert. Wenn Erkenntnisse bei der Auswertung der Interviews mit Probanden, oder der Nutzerumfrage von Saros gewonnen wurden, ist dies ebenfalls gekennzeichnet.

1	**Kategorie Awareness**	**Titel: Keine Anzeige welcher Nutzer wem folgt**	**severe/schwer**	**4**

Es ist nicht ersichtlich ob und wann Nutzer A einem Nutzer B folgt. Nutzer müssen verbalisieren und fragen die Information wiederholt ab ("Folgst du mir?","Wo bist du gerade?")
Heuristics: H1 Sichtbarkeit des Systemstatus, H6 Erkennen vor Erinnern
Trigger: HE + UT + UT[Solovjev2011] - Ständiges verbalisieren der Probanden.
Typ: Nutzer fragte nach Hilfe
Qualifier: Missing (M)

Änderung: FollowMode-Activitites werden von allen Nutzern an alle Nutzer gesendet. Anpassung des Saros-Views, sodass die Baumelemente der Sitzungsteilnehmer Kindelemente haben, die entsprechende Informationen darstellen

2	**Kategorie Awareness**	**Titel: Welche Datei hat Nutzer X offen unklar**	**severe/schwer**	**4**

Saros nutzt nur sehr kleine, von Nutzern nicht wahrgenommene Icon-Annotationen von (x*x Pixeln) mit kleinen gelben und grünen Kreisen z.B. im Package Explorer um anzuzeigen, welche Datei ein Nutzer offen hat. Dies ist für Sitzungen mit mehr als 2 Teilnehmern ungeeignet, da hier nicht codiert ist, wer die Datei offen hat. Auch ist die Anzeige unzuverlässig, da die übergeordneten Ordnerstrukturen nicht gekennzeichnet werden, und die Markierung somit unsichtbar ist, solange nicht gerade der Ordner offen ist, in dem die geöffnete Datei liegt.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: HE + UT + UT[Solovjev2011]
Typ: Eingeschränkte Effizienz

[30] http://blog.net-corps.de/diplom/

Qualifier: Missing (M)

Änderung: Saros-View wurde ergänzt: Unterhalb der Baumelemente für Sitzungsteilnehmer ist nun ein weiterer Eintrag der anzeigt, ob ein Nutzer eine geteilte Datei offen hat, und den Pfad der Datei im Format "Projektname: OrdnerA/OrdnerB/Datei.java" Ein Doppelklick hierauf löst die "Jump to user position" Aktion aus.

3	Kategorie Awareness	Titel: Nutzer weiß nicht ob ihm gefolgt wird	severe/schwer	4

Saros liefert Nutzern keine Informationen darüber, ob andere Nutzer ihnen folgen. Es gibt keine Indikation darüber, ob ein anderer Teilnehmer im Follow-Modus ist.
Heuristics: H1 Sichtbarkeit des Systemstatus, H6 Erkennen vor Erinnern, H10 Hilfe und Dokumentation
Trigger: UT [Solojev2011] + eigene UTs + HE
Qualifier: Missing (M)
Persistenz: Repeatedly

Änderung: Entsprechende Datenübertragung wurde implementiert. Der Saros-View wurde angepasst um diese Informationen darzustellen.

4	Kategorie Awareness	Titel: Keine Information über Verlust der Verbindung in Sitzung	severe/schwer	4

Wenn Nutzer die Verbindung verlieren in einer Sitzung sieht es in Saros so aus als würde die Sitzung weiterlaufen. Keine Indikation das Verbindung verloren und Sitzung nicht mehr aktiv.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: Umfrage - Nutzer entwickelten 5 Minuten weiter ohne zu merken das die Sitzung "tot" ist, danach manuelles "mergen" der Änderungen, da changelog fehlt, usw...
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)

Lösungsansatz: PingPongAktivitäten existieren bereits: überabeiten und Nutzer informieren das Verbindung zusammengebrochen + Sitzung beenden. Ideal: Sitzung wiederaufnehmen und Synchronisieren (aktuell wohl unmöglich mit der Saros-Architektur)

5	Kategorie Awareness	Titel: Keine offensichtliche Indikation der Host-Rolle	moderate/mittel	3

Nutzer sehen keine visuelle Indikation, wer in einer Sitzung der Host ist. Dies führte zu Verunsicherung und Spekulationen: z.B. vermutete ein Nutzer, dass der Host immer Pink ist, da der Host in einer vorherhigen Sitzung diese Farbe hatte. Auch war Nutzern unklar, wer nun eine Session beenden darf bzw. sollte, und versuchten zu ermitteln, wer der Host ist, damit dieser die Sitzung beendet.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H6 Erkennen vor Erinnern

Trigger: UT - Verbale Koordination, wer die Funktion zum Ändern der Rechte nutzen kann

Änderung: Anzeige der Host-Rolle vor dem Namen des Nutzers im Saros-View.

6	Kategorie Awareness	Titel: Scheinbar willkürliche Sortierung der Sitzungs-Teilnehmer	moderate/mittel	3

Die Sortierung der Sitzungsteilnehmer wirkt unharmonisch, da der eigene Eintrag "You" mal hier und mal dort angezeigt wurde.
Heuristics: H4 Konsistenz und Standards
Trigger: UT - Verzögerungen durch langsame Orientierung im Saros-View Die Sortierung hat sich den Nutzern nicht erschlossen, so dauerte es länger den Eintrag des jeweiligen Buddies zu suchen.
Typ: Eingeschränkte Effizienz

Änderung: Sortierung fixiert: - You ist immer am Ende der Liste, da wenigste Interaktionswahrscheinlichkeit - Sortierung alphabetisch

7	Kategorie Awareness	Titel: Zufällige Farbverteilung / Kein Merken der Farbkonfigurationen	moderate/mittel	3

Aus den 5 verfügbaren Farben werden die Farben scheinbar zufällig zugeordnet. Bei 2er-Sitzungen meist wechselweise, sodass Nutzer verwirrt sind, wieso in der zweiten Sitzung der Nutzer A plötzlich die Farbe des Nutzers B aus der ersten Sitzung hat. Farbkonfigurationen werden nicht gespeichert, d.h. bei unpassenden/unerwünschten Farben muss der Nutzer in jeder Sitzung seine Farbe neu konfigurieren.
Heuristics: H3 Benutzerkontrolle und Freiheit, H4 Konsistenz und Standards, H8 Ästhetisches und minimalistisches Design
Trigger: UT - Nutzer ist überzeugt, Host sei immer Pink (sic!) durch unverständliches Systemverhalten bei Farbwahl
Qualifier: Incongruent mental model (I)

Lösungsansatz: Saros sollte sich die Farbkombinationen merken, und insbesondere versuchen einem Nutzer, der seine Farbe explizit umgestellt hat, auch in einer folgenden Sitzung wieder diese Farbe zuzuweisen.

8	Kategorie Awareness	Titel: Annotationen für Projekte die Bestandteil einer Sitzung sind, sind zu subtil (winzige Icons an Icons)	severe/schwer	3

Die Markierungen an Projekten und Dateien die Bestandteil einer Sitzung sind, sind sehr subtil (winzige Icon-Annotationen an Icons).
Heuristics: H1 Sichtbarkeit des Systemstatus, H6 Erkennen vor Erinnern, H8 Ästhetisches und minimalistisches Design
Trigger: HE + UT + UT[Kahlert2011] - Nutzer suchen zeitintensiv ob das Projekt Bestandteil der

Sitzung ist, es genügt nicht "ein Blick" um festzustellen welche Projekte synchronisiert sind.
Typ: Nutzer fragte nach Hilfe
Qualifier: Overlooked (O)

Änderung: Suffix-Dekoration an Projekte angefügt (share) bzw. (partial share).

9	Kategorie Awareness	Titel: Cursor-Position nicht deutlich genug (Zeilenindikation)	moderate/mittel	3

Die Hervorhebung der aktuellen Cursor-Position ist schwach. Die Cursor-Position innerhalb einer Markierung durch einen anderen Nutzer, oder die Markierung in Code, der durch den aktuellen Nutzer kürzlich geändert wurde und somit hervorgehoben ist, hebt sich nicht ausreichend ab. Klickt ein Nutzer in eine Code-Zeile in der kein Text steht, ist zudem gar nicht zu erkennen, wo der Cursor sich befindet ist.
Heuristics: H8 Ästhetisches und minimalistisches Design
Trigger: Nutzer wünschten sich in Interviews eine Art Laserpointer, oder allgemein eine stärkere Hervorhebung der Cursorposition
Typ: Eingeschränkte Effektivität
Qualifier: Better way (B)

Lösungsansatz: Bessere Indikation der Cursor-Position durch farbigen Pfeil in der Seitenleiste des Editors wäre eine Idee. Auch die gesamte Zeile hervorzuheben, in der sich der Cursor befindet (so wie es der Nutzer schon von Eclipse selbst kennt), wäre ein Möglichkeit.

10	Kategorie Awareness	Titel: Unzuverlässige Markierung der Dateien die synchronisiert sind	moderate/mittel	3

Die Markierung mit dem blauen Doppelpfeil an synchronisierten Dateien erfolgt oft nur teilweise (nicht für alle Ressourcen). Abhilfe schafft hier nur ein manueller Refresh der Projekte durch den Nutzer. Caching-Probleme in der Baumdarstellung durch Eclipse
Heuristics: H1 Sichtbarkeit des Systemstatus, H6 Erkennen vor Erinnern
Trigger: HE + UT

Lösungsansatz: Ermitteln, wie ein zeitnaher Refresh der Baumdarstellung erfolgen kann.

11	Kategorie Awareness	Titel: Keine Indikation der Mausposition	moderate/mittel	3

Saros bietet keine Möglichkeit, ohne explizit in den Code zu klicken, bzw. texttuell zu markieren auf etwas zu zeigen - es gibt keine Indikation der Mausposition.
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: UT - Feature-Wunsch, Nutzer erläutern ohne zu klicken und zielen mit Maus auf Code-Teil den

sie kommentieren
Typ: Eingeschränkte Effektivität
Qualifier: Incongruent mental model (I)

Lösungsansatz: Ein Telepointer (eine Art Laserpointer der den Mauszeiger repräsentiert) wäre eine Möglichkeit. Es wäre sogar denkbar, eine Annotation-Ebene (wie eine Folie) auf jeden synchronisierten Editor zu legen, auf der wie in einem Whiteboard gemalt und geschrieben werden kann.

12	Kategorie Awareness	Titel: Klick-Event wird nicht visualisiert	moderate/mittel	3

Klick-Events werden von Saros nicht visualisiert. Lediglich die Cursor-Position und Textmarkierungen, sowie Textänderungen werden übertragen.
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: UT[Solovjev2011] + eigene UTs Nutzer klicken wiederholt auf dieselbe Stelle Code, um deutlich zu machen, worüber sie gerade erzählen.
Typ: Eingeschränkte Effektivität
Qualifier: Missing (M)
Persistenz: Repeatedly

Lösungsansatz: Klickevents visualisieren. Es wurde spontan keine Möglichkeit gefunden dies mit den von Eclipse bereitgestellten Annotations zu realisieren.

13	Kategorie Awareness	Titel: Keine visuelle Indikation welche Viewparts synchronisiert	moderate/mittel	3

Saros visualisiert nicht, welche Viewparts synchronisiert werden
Heuristics: H1 Sichtbarkeit des Systemstatus, H6 Erkennen vor Erinnern
Trigger: UT - Nutzer kommentieren Consolen-Ausgabe, oder markieren / kommentieren Anzeigen in Editoren, die nicht synchronisiert werden. Frustration & zeitliche Verzögerung darüber (wenn verbale Koordination dazu erfolgen muss ,um dieses Problem zu erkennen)
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Overlooked (O)

Lösungsansatz: Farbiger dicker Rahmen um Views, die synchronisiert werden, so ähnlich wie bei Screnesharingprogrammen, die den übertragenen Teil des Bildschirms einrahmen

14	Kategorie Awareness	Titel: Meldung bei Inkonsistenz zu subtil	moderate/mittel	2

Die Meldung das eine Inkonsistenz aufgetreten ist, ist zu subtil und gleichzeitig zu lang, wodurch die Nutzer wenig motiviert waren, die gesamte Meldung zu lesen und zu verstehen. Da die Balloon-

Benachrichtigungen sich nach einer bestimmten Zeit (in Abhängigkeit der Länge des angezeigten Textes, sowie einer Mindestdauer) automatisch schließen, geht diese Information verloren. Zudem kann die Aktion "Inkonsistenz auflösen" nicht ausgelöst werden durch Anklicken der Benachrichtigung. Hierzu muss zunächst das Icon gefunden werden, das in der Meldung beschrieben wird. Auch lässt die Dokumentation der Inkonsistenz-Situation zu wünschen übrig. So ist weder ausreichend dokumentiert, wie es zu einer Inkonsistenz kommen konnte, noch
Heuristics: H4 Konsistenz und Standards, H5 Fehler vermeiden, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern, H10 Hilfe und Dokumentation
Trigger: UT - Nutzer reagierten positiv auf das Erkennen der Inkonsistenz (wurde als Robustheitseigenschaft des Systems interpretiert), da sie beim Auftreten der Meldung allerdings nicht gewzungen sind zu agieren , und das Inconsistency-Icon für die Nutzer offenbar zu subtil ist, ignorieren sie die Meldung und arbeiten weiter. Wenn Nutzer eine Entscheidung nicht treffen müssen, verzichten sie meist lieber darauf, da dies eine geringere kognitive Belastung darstellt.
Typ: Nutzer fragte nach Hilfe
Qualifier: Incongruent mental model (I)

Lösungsansatz: Da unklar ist ob die Inkonsistenz negative Auswirkungen haben wird auf die weitere Arbeit in der Sitzung kann nicht klar bestimmt werden, ob

15	Kategorie Awareness	Titel: Icon Annotation für "Eclipse-Fenster im Hintergrund" unklar	minor/leicht	1

Das gewählte Icon das Nutzer innerhalb des Saros Views markiert (eine Uhr) ist ungeeignet um darzustellen, das dieser Nutzer das Eclipse Fenster nicht aktiv fokussiert hat. Durch Anordnung von Fenstern auf dem Monitor, oder Mehrbildschirm-Konfigurationen ist diese Einstellung weitestgehend aussagelos. Zudem hat keiner der Probanden verstanden was dieses kleine Icon bedeuten soll, da es nicht geeignet dokumentiert ist
Heuristics: H1 Sichtbarkeit des Systemstatus, H8 Ästhetisches und minimalistisches Design, H10 Hilfe und Dokumentation
Trigger: HE+UT
Qualifier: Overlooked (O)

16	Kategorie Einladungsprozess	Titel: Immense Verzögerung bevor sich der Einladungswizard öffnet	severe/schwer	4

Wenn in einem Package Explorer artigen View etwas selektiert ist (in irgendeiner Perspektive, in irgendeinem Navigator-artigen View), wird die Selektion übernommen. Dies ist zeitintensiv und verhindert die unmittelbare Anzeige des Wizards.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: HE + UT - Nutzer sind nicht sicher, ob die Aktion "Share with multiple buddies" oder "Share Project(s)" erfolgreich angeklickt und ausgelöst wurde, oder ob sie den Menüpunkt schlicht verfehlt haben. Nutzer versuchen erneut eine Einladung und wundern sich, dass die Projekte nicht mehr selektierbar sind (da sie für Saros bereits als Bestandteil der gestarteten Sitzung sind).

Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)
Persistenz: Permanently

Änderung: Änderungen in der Programmlogik (Asynchrone Übernahme der Selektion) führt zu sofortiger Darstellung des Wizards.

17	Kategorie Einladungsprozess	Titel: Ressourcen-Selektion wird aus allen Perspektiven aggregiert	severe/schwer	4

Wenn in irgendeiner Perspektive, in irgendeinem Navigator-artigen View etwas selektiert ist (Datei/Ordner/Projekt), wird diese Selektion übernommen beim Starten des Einladungs-Wizards. Hierbei werden Selektionen übernommen, die für den Nutzer gar nicht sichtbar sind (z.B. aus der Debug-Perspektive) - so werden Projekte versendet, ohne dass der Host dies bemerkt, da es vor dem Absenden der Einladung keine Zusammenfassung gibt, welche Projekte jetzt versendet werden (und wie groß diese sind). Dies kann schwerwiegende Folgen haben, wenn z.B. ungefragt Projekte verschickt werden, deren Code nicht weitergegeben werden darf an den eingeladenen Entwickler.
Heuristics: H3 Benutzerkontrolle und Freiheit, H5 Fehler vermeiden, H6 Erkennen vor Erinnern
Trigger: UT - Nutzer beschwerte sich, wieso Projekte übertragen wurden, die nicht synchronisiert werden sollten, da diese nicht Bestandteil der Sitzung waren.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)
Persistenz: Permanently

Änderung: Behebung des Programmfehlers in Zusammenarbeit mit einem Teamkollegen. Hinzufügen einer Zusammenfassung mit Liste aller zu sendenden Projekte, als neue letzte Seite des Einladungswizards.

18	Kategorie Einladungsprozess	Titel: Mangelnde Performance bei Einladen mehrerer Nutzer	severe/schwer	4

Dadurch, dass pro Nutzer redundant Checksummen berechnet werden, ist der Einladungsprozess zu Beginn extrem langsam, gerade bei HDDs (bei SSDs geht es noch), besonders durch parallelen Festplattenzugriff auf identische Dateien.
Heuristics: H1 Sichtbarkeit des Systemstatus, H7 Flexibilität und effiziente Nutzung
Trigger: UT - Frustration über langlaufenden Einladungsprozess. Zusammenfassen der Jobs aus einem Plugin wurde als selbstverständlich angesehen.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Better way (B)
Persistenz: Permanently

Änderung: Checksummen werden zwischengespeichert. Die technische Umsetzung erfolgte nach Melden des Problems im Bugtracker durch einen Teamkollegen.

19	Kategorie Einladungsprozess	Titel: Speicherverbrauch ist enorm	severe/schwer	4

Dadurch das die ZIP-Dateien für das Versenden der Projekte pro eingeladenem Nutzer zunächst komplett in den Speicher geladen werden kommt es in Abhängigkeit von Systemkonfiguration und Hardwareausstattung sehr schnell zu OutOfMemoryExceptions (bzw. Java Heap Space) > Datenverlust!
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: Saros-Entwickler-UTs - Abstürze von Eclipse bei Tests mit großen Projekten.
Typ: Eingeschränkte Effizienz
Persistenz: Permanently

Lösungsansatz: Streaming-Lösung, bei der die ZIP-Datei schrittweise eingelesen werden beim Sendeprozess. Erste Ansätze einer Implementierung für einen Streaming-Service existieren bereits, allerdings ist diese Lösung momentan nicht mehr funktionsfähig und nach Aussagen von Teamkollegen ungeeignet implementiert worden.

20	Kategorie Einladungsprozess	Titel: Kein Feedback bei der Berechnung von Dateichecksummen	severe/schwer	4

Das System gibt dem Nutzer kein Feedback über den Fortschritt der Berechnung von Checksummen (ein lang laufender Prozess, mit vielen Festplattenzugriffen). Es gibt keine Fortschrittsanzeige, und aufgrund der Implementierung war es nicht möglich, die Anzahl der zu verarbeitenden Dateien für einen Fortschrittsbalken zu nutzen.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: HE + UT + UT[Solovjev2011] - Nutzer äußern sich wiederholt negativ über mangelnde Performance und völlig unklares Systemverhalten
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Better way (B)
Persistenz: Permanently

Änderung: Die Dateien werden zuerst komplett aufgelistet, sodass ein geeigneter Fortschrittsbalken mit nahezu konstanter Bewegung dargestellt werden kann. Es werden detaillierte Informationen über den Berechnungsfortschritt angezeigt, inklusive der Dateinamen. Dies ermöglicht es Nutzern frühzeitig zu erkennen, das ggf. große, nicht notwendige Dateien verarbeitet werden (die selektiert wurden).

21	Kategorie Einladungsprozess	Titel: Scan Workspace langwierig ohne Feedback,nicht abbrechbar, undokumentiert	severe/schwer	4

Das Scannen des Workspace nach passenden Projekten anhand der Dateilisten mit Checksummen dauert je nach Umfang der Projekte im Workspace sehr lange: - Es gibt kein Feedback - Prozess ist nicht abbrechbar - Funktionalität ist nicht dokumentiert und missverständlich. Scan Workspace wählt das erste Projekt, das mit mehr als 80% übereinstimmt mit dem ankommenden Projekt. Damit erfüllt es nicht den eigentlichen Sinn und Zweck: Das Projekt finden das am ehesten passt, da der

Scanvorgang beim ersten Treffer mit mehr als 80% sofort abbricht. Ein besser passendes Projekt wird so ignoriert. Auch ist für den Nutzer nicht dokumentiert, dass die Funktion nach einem 80%igen Match sucht.

Heuristics: H1 Sichtbarkeit des Systemstatus, H3 Benutzerkontrolle und Freiheit, H8 Ästhetisches und minimalistisches Design, H10 Hilfe und Dokumentation

Trigger: HE + UT + UT [Solovjev2011]

Typ: Eingeschränkte Effektivität

Qualifier: Incongruent mental model (I)

Persistenz: Permanently

Änderung: Die Funktionalität wurde nach reiflicher Überlegung entfernt, da offenbar keine sorgfältige Anforderungserhebung für diese Funktionalität vorgenommen wurde. Die aktuelle Implementierung ist ungeeignet für den Haupt-Einsatzzweck und wurde entfernt, was zugleich die Übersicht des Wizards deutlich verbessern konnte. Tatsächliche Anwendungsfälle für diese Funktionalität konnten keine hergeleitet werden.

22	Kategorie Einladungsprozess	Titel: 60 Sekunden Verzögerung ohne Feedback durch synchronen Chatraum-Aufbau	severe/schwer	4

Hat der Chatserver technische Probleme, gibt es bis zu 60 Sekunden Verzögerung ohne Feedback durch synchronen Chatraum-Aufbau.

Heuristics: H1 Sichtbarkeit des Systemstatus, H5 Fehler vermeiden

Trigger: HE (Eigene Testläufe)

Änderung: Asynchrones Starten des Verbindungsaufbaus für den Chatraum

Lösungsansatz: Hinweis muss noch angezeigt werden, der Nutzer mitteilt das der Chatraum nicht geöffnet werden konnte, wegen eines technischen Problems

23	Kategorie Einladungsprozess	Titel: Große Verzögerung zwischen Senden und Empfangen der Einladung	severe/schwer	4

Der Host sendet eine Einladung, und teilt dies auch verbal mit, allerdings sieht der Empfänger die Einladung erst sehr viel später, da zunächst beim Host Checksummen für alle Dateien berechnet werden, was im Falle des Ablehnens der Einladung durch den eingeladenen Nutzer reine Zeitverschwendung war.

Heuristics: H1 Sichtbarkeit des Systemstatus

Trigger: HE + UT - Verwirrung über Verzögerung zwischen Senden und Empfangen der Einladung, dadurch erneuter Versuch einer Einladung.

Typ: Nutzer äußert Frustration (verbal)

Änderung: Reihenfolge der technischen Implementierung überarbeitet. Checksummenberechnung frühestens wenn der eingeladene Nutzer die Einladung angenommen hat. Checksummen-Berechnung mit Caching-Mechanismus für Beschleunigung des Prozesses

und vermeiden redundanter Berechnungen.

24	Kategorie Einladungsprozess	Titel: Kein Feedback über Dauer des Einladungsprozesses	severe/schwer	4

Der Host und auch eingeladene Nutzer haben keine Möglichkeit abzuschätzen, wie lange ein Einladungsprozess noch dauern wird. Die Fortschrittsbalken bewegen sich nur sprunghaft zu fest eingestellten Prozentwerten und zeigen keine Übertragungsgeschwindigkeit an und auch nicht wieviele Daten noch versendet werden müssen.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: HE + UT + UT[Solovjev2011]

Änderung: Umfassende Änderungen der Fortschrittsanzeigen, Ergänzung um Übertragungsgeschwindigkeit, zu versendende Datenmenge, versendete Datenmenge, noch verbleibende Datenmenge, und erwartete Restlaufzeit

25	Kategorie Einladungsprozess	Titel: Keine initiale Rechtevergabe möglich	moderate/mittel	3

Will man, dass ein Nutzer der eingeladen wird, nichts am Projekt verändern kann, muss man diesem die Schreibrechte entziehen. Dies geht nur durch den Host und nur über das Kontextmenü. Ist der Host nicht schnell genug mit der Umstellung, besteht die Möglichkeit, dass der eingeladene Nutzer bereits etwas modifiziert hat, bevor ihm die Rechte entzogen werden. Zudem ist die Funktion im Kontextmenü so versteckt, das sie nicht jedem Nutzer auffällt.
Heuristics: H3 Benutzerkontrolle und Freiheit
Trigger: HE + UT - Option im Kontextmenü wurde nicht entdeckt/genutzt
Qualifier: Overlooked (O)

Änderung: Option im Einladungswizard bei der Buddy-Selektion integriert

26	Kategorie Einladungsprozess	Titel: Der Mauszeiger flackert wild bei Ressourcen-Selektion	moderate/mittel	3

Der Mauszeiger flackert zwischen normal-Zustand und Busy-Indikation (z.B. die Sanduhr bei Windows) während die Selektionen aus den Navigator-Views übernommen werden in den Wizard.
Heuristics: H8 Ästhetisches und minimalistisches Design
Trigger: Dies wirkte wie ein Defekt auf den Nutzer.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)
Persistenz: Permanently

Änderung: Konsequente Verwendung des BusyIndicator sorgt für konstante Darstellung eines

Mauszeigers der Aktivität repräsentiert.

27	Kategorie Einladungsprozess	Titel: Create copy for working distributed defekt und unklar	moderate/mittel	3

Die Funktion "Create copy for working distributed" ist nicht dokumentiert, und dem Nutzer unklar. Zudem ist sie defekt und erfüllt nicht einmal den einzigen Zweck den sie hat.
Heuristics: H4 Konsistenz und Standards
Trigger: HE + UT[Kahlert2011]
Persistenz: Permanently

Änderung: Die Funktionalität ist anders realisiert worden und behebt dabei auch den Fehler. Nutzer können für das lokale Ablegen eingehender Projekte nun ein existierendes Projekt des Workspaces aus einem Dropdown wählen und mit einer Schaltfläche mit softortiger Wirkung eine Kopie anlegen. Angleichung an Eclipse Verhalten und Nutzererwartung

28	Kategorie Einladungsprozess	Titel: Disable version Control support auch für Projekte ohne SVN Bindung angeboten	moderate/mittel	3

Die Option ist auch verfügbar, wenn keines der eingehenden Projekte eine SVN Bindung hat (somit hat die Option keinerlei Effekt)
Heuristics: H4 Konsistenz und Standards
Trigger: HE

Änderung: Durch komplette Restrukturierung des Wizards kann die Option nun pro Projekt gewählt werden und wird auch nur angeboten, wenn sie überhaupt verfügbar ist (da eine SVN-Bindung am Projekt gegeben ist).

29	Kategorie Einladungsprozess	Titel: TabControl für eingehende Projekte unergonomisch	moderate/mittel	3

Das TabControl ist so gut wie unbedienbar für mehr als 5 Projekte. Zudem haben diverse Nutzer die Tabs nicht entdeckt, bestätigen direkt den Wizard-Dialog mit Finish und legen bei jeder Sitzung alle Projekte neu ab. Dies dauert damit unnötig lange.
Heuristics: H4 Konsistenz und Standards, H8 Ästhetisches und minimalistisches Design
Trigger: HE + UT: Nutzer bemerken nicht, dass sie gerade mehrere Projekte erhalten, und wie sie diese konfigurieren können.
Persistenz: Repeatedly

Änderung: Veränderung des Wizards auf Listendarstellung und Verringerung der verfügbaren

Optionen auf das nötigste.

30	Kategorie Einladungsprozess	Titel: Keine Session-Resume-Funktionalität	moderate/mittel	3

Es gibt keine Möglichkeit, eine unterbrochene Sitzung schnell fortzusetzen. Nutzer müssen manuell Ressourcen selektieren

Heuristics: H7 Flexibilität und effiziente Nutzung

Trigger: Feature-Wunsch aus Saros-Nutzerumfragen / Interviews

Typ: Eingeschränkte Effizienz

Qualifier: Missing (M)

Persistenz: Permanently

Änderung: Ein schneller Neustart der Sitzung ist möglich, durch leichtes wiederherstellen der letzten Projektselektion

Lösungsansatz: Weitere Optimierungsmöglichkeiten wäre eine "Resume Last Session" Schaltfläche die automatisch Einladungen auslöst für die Teilnehmer der letzten Sitzung. Ideal: Automatische Verwendung der zuletzt zugewiesenen Farben

31	Kategorie Einladungsprozess	Titel: Keine Funktionalität um getrennte Änderungen zusammenzuführen	moderate/mittel	3

Haben Entwickler getrennt an einem gemeinsamen Projekt gearbeitet, gibt es keine Möglichkeit, wie z.B. von SVN merge bekannt, die Projektstände zusammenzuführen. Die Änderungen von einem der beiden Nutzer wird zwangsläufig überschrieben

Heuristics: H7 Flexibilität und effiziente Nutzung

Trigger: Nutzerumfragen: Feature-Requests

Lösungsansatz: Strukturelles Problem - SVN Merge implementieren sehr aufwändig (wenn überhaupt machbar). Kein Lösungsansatz gefunden

32	Kategorie Einladungsprozess	Titel: SVN-Revisionen werden immer ermittelt	moderate/mittel	3

Dadurch das beim Host die SVN-Revisionsnummern von allen Dateien in Projekten die geteilt werden sollen, und eine SVN-Bindung haben ermittelt werden, dauert der Einladungsprozess deutlich länger. Wenn nun der eingeladene Nutzer SVN grundsätzlich deaktiviert hat in seinen Einstellungen war dies vollkommen überflüssig.

Heuristics: H3 Benutzerkontrolle und Freiheit, H7 Flexibilität und effiziente Nutzung

Trigger: HE - Zeitverschwendung durch unnötig langlaufenden Einladungsprozess

Lösungsansatz: Bei der Session-Aushandlung Information senden ob Nutzer überhaupt SVN-Nutzung aktiviert hat.

33	**Kategorie Einladungsprozess**	**Titel: Bedeutung der Datei-Annotationen (Icon-Überlagerungen) sind unklar**	**moderate/mittel**	**3**

Zur Prüfung des erfolgreichen Sitzungsaufbaus suchen Nutzer nach einer Indikation das die Projekte erfolgreich synchronisiert wurden. Die Icon-Annotationen (blauer Doppelpfeil) sind für den Nutzer schwer zu erkennen und schwer zu intepretieren.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: UT[Kahlert2011]
Typ: Eingeschränkte Effektivität
Qualifier: Overlooked (O)

Änderung: Durch Hinzufügen von textuellen Suffixen an die Projektnamen in Navigator-Views: "(share)" und "(partial share)" erkennen Probanden nun sofort, das die Projekte erfolgreich zur Saros Sitzung hinzugefügt wurden. Die Icons bleiben nach wie vor unerklärt für den Nutzer.

Lösungsansatz: Das Tutorial ("Getting started" Webseite) sollte Nahaufnahmen der Markierungen in Screenshots zeigen um den Nutzern klar zu machen, welche Bedeutung die Markierungen haben, und wie diese von Nutzern verwendet werden können.

34	**Kategorie Einladungsprozess**	**Titel: Verwirrung Verhalten von "Create copy for working distributed"**	**moderate/mittel**	**3**

Das Verhalten von "Create copy for working distributed" ist schlecht dokumentiert und damit verwirrend.
Heuristics: H10 Hilfe und Dokumentation
Trigger: UT[Kahlert2011] Sess-17 Nutzer ging davon aus, das der dort eingegebene Name sein Backup wird und nicht seine Arbeitskopie.
Persistenz: Permanently

Änderung: Die Funktionalität wurde gänzlich anders implementiert - Kopien werden nun sofort angelegt, wodurch Nutzer direkt erkennen können, das die Kopie als Arbeitsgrundlage verwendet wird, und nicht das Original.

35	**Kategorie Einladungsprozess**	**Titel: Keine Koordination mit Saros außerhalb einer laufenden Sitzung möglich**	**moderate/mittel**	**2**

Chat, VoIP und Screensharing sind allesamt gekoppelt an eine laufende Sitzung. Durch Änderungen an den Wizards ist es nicht mehr möglich, Sitzungen ohne Projekte zu starten. Nutzer können so weder per

Chat, noch via VoIP koordinieren, wann alle Teilnehmer Zeit für eine Sitzung haben. Es ist nicht möglich, anzugeben, was genau gemacht werden soll in einer Sitzung - die Koordination kann erst in der laufenden Sitzung erfolgen. Nutzer tendieren allerdings generell dazu eine IM oder VOIP-Applikation zu nutzen und darüber Termine für Sitzungen zu koordinieren, weshalb es "nur" unbequem ist (Nutzer würden sich allerdings eine vollständige Integration wünschen). Es gibt keine Möglichkeit anzugeben, wozu man einlädt oder wieso jemand ablehnt.
Heuristics: H4 Konsistenz und Standards
Trigger: HE + UT
Qualifier: Missing (M)
Persistenz: Once

Lösungsansatz: Es gibt bereits Ansätze für die Implementierung eines persönlichen Chats zwischen zwei Nutzern, unabhängig von einer laufenden Sitzung. Eine mögliche Alternative dazu wäre gewesen, bei der Einladung einen freien Nachrichtentext mitsenden zu können, sodass der einladende Nutzer angeben kann, wieso gerade eingeladen wird. Der Nachteil hierbei wäre, dass der eingeladene Nutzer nun auch eine Antwortmöglichkeit bräuchte, womit man schon wieder bei einer Chatfunktionalität wäre.

36	Kategorie Einladungsprozess	Titel: Feature / Keine Liste von laufenden Sitzungen verfügbar	minor/leicht	2

Es gibt keine Ansicht welche Nutzer sich in einer Sitzung befinden, und keine Möglichkeit eigenständig einer Sitzung beizutreten oder die Nutzer zu kontaktieren und zu fragen ob man mitmachen kann.
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: UT - Nutzer kennt Software "Xpairtise", das das freie Beitreten zu laufenden Sitzungen ermöglicht und nennt dies als mögliche neue Funktion
Typ: Eingeschränkte Effektivität
Qualifier: Better way (B)

Lösungsansatz: Dies ist eine konzeptionelle Entscheidung in Saros, die möglicherweise nicht explizit getroffen wurde. Eine derartige Funktion zu implementieren dürfte möglich sein, und wäre eine gute Ergänzung für Entwickler die in größeren Teams arbeiten (Workspace/Virtuelles großes Büro), und wechselweise zusammenarbeiten. Fraglich ist, ob der Aufwand den Nutzen rechtfertigen würde. Hier wären weitere Untersuchungen nötig.

37	Kategorie Einladungsprozess	Titel: Nutzer ohne Saros-Unterstützung und Offline-Nutzer selektierbar	moderate/mittel	2

Nutzer die nicht mit Saros in ihrem XMPP-Account eingeloggt sind, oder offline sind sind selektierbar. Wird ein solcher Buddy selektiert, kann der Nutzer den Wizard nicht abschließen bis die Selektion entfernt wurde.
Heuristics: H5 Fehler vermeiden, H7 Flexibilität und effiziente Nutzung
Trigger: Buddylisten können sehr lang sein, und unübersichtlich. Dies erschwert das schnelle Durchsuchen der Buddyliste nach den Kontakten die man einladen möchte.

Änderung: Änderung der Standard-Einstellung, sodass Nutzer, die offline sind oder online, aber nicht Saros verwenden, initial nicht dargestellt werden. Der Nutzer kann sich aber über eine Checkbox vergewissern, das der Nutzer in seiner Buddyliste existiert, aber nicht online ist.

Lösungsansatz: Es erscheint etwas überflüssig, überhaupt Buddies anzeigen zu können, die nicht eingeladen werden können für eine Sitzung. Allerdings müsste dann eine Aktualisieren-Schaltfläche integriert werden, die die Liste aktualisiert für den Fall das ein Buddy, der beim Öffnen des Wizards offline war, und später online gegangen ist, in der Liste auftaucht. Es existiert offenbar noch ein Fehler der in einigen (nicht näher bestimmten) Fällen dafür sorgt, dass die Liste sich nicht aktualisiert. Durch Nutzen der Checkbox kann an-& abschalten der Checkbox kann der Nutzer dieses Problem umgehen.

38	Kategorie Einladungsprozess	Titel: Keine Zusammenfassung, welche Resouren verschickt werden (inkl. Datenmenge)	minor/leicht	2

Der Host hat keine Übersicht, wie groß die Datenmenge ist, die versandt wird, und welche Projekte er selektiert hatte.
Heuristics: H6 Erkennen vor Erinnern
Trigger: UT - Nutzer war verwirrt wieso andere Projekte verschickt wurden, als er bewusst ausgewählt hatte. Übersicht hätte ggf. geholfen das Problem zu lokalisieren.
Persistenz: Permanently

Änderung: Anzahl der selektierten Projekte wird angezeigt bei der Projektselektion.

Lösungsansatz: Die Ermittlung der Dateigröße führt zu einer Verlangsamung des gesamten Einladungsprozesses, da die Informationen schwer zwischengespeichert werden können in der jetzigen Saros-Klassenstruktur. Würde der Zugriff auf Systemressourcen weiter gekapselt könnten hier ein Cache (Zwischenspeicher) eingeführt werden, der die Geschwindigkeit sogar verbessern könnte, und nicht nur gleichbleibend lässt.

39	Kategorie Einladungsprozess	Titel: Überlappende Fortschrittsbalken bei Einladen mehrere Nutzer	minor/leicht	2

Die Fortschrittsbalken für den Einladungsprozess jedes Nutzers liegen direkt übereinander. Die hinteren sind damit unsichtbar, bis der Nutzer einen minimiert oder verschiebt. Fortschrittsbalken sind nicht gruppiert.
Heuristics: H4 Konsistenz und Standards
Trigger: UT[Solovjev2011] - Nutzer sind genervt, dass sie mehrere Fortschrittsbalken minimieren müssen über die Schaltfläche "Run in Background" und die Balken überlappen sich, wodurch für den Host Informationen fehlen. Allerdings waren die Nutzer in der Lage, schnell zu erkennen, das mehrere Fortschittsbalken hintereinander liegen, da sie die Fortschittsbalken mit der Maus in ihrer Position verschoben
Typ: Eingeschränkte Effektivität
Qualifier: Better way (B)

Änderung: Durch Anpassung der Job-Familie (Überschreiben der getFamily() Methode in den Jobs) ist Eclipse nun in der Lage die Jobs unter "Details" in jedem der Fortschittsdialoge darzustellen (als Liste). [Screenshot]

Lösungsansatz: Es wäre schön, wenn alle Jobs zusammengefasst werden könnten, die mit dem Einladungsprozess von Saros zu tun haben. Leider war dies bei entsprechenden Tests nicht ohne weiteres möglich, durch die Trennung der Bestandteile einer Sitzungseinladung und führte zu weiteren Problemen: Einzelne Einladungsprozesse konnten nun nicht mehr abgebrochen werden, d.h. bei Einladen mehrer Nutzer konnte der Host nun nur noch den gesamten Einladungsprozess abbrechen, und nicht

40	Kategorie Einladungsprozess	Titel: "Disable version control support" Option unklar	minor/leicht	2

Negative Formulierung ist überraschend und die Option ist schwer verständlich. Da sie SVN spezifisch ist, macht es wenig Sinn diese abstrakt zu bezeichnen.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H4 Konsistenz und Standards
Trigger: HE

Änderung: Checkbox umbenannt mit Formulierung, die SVN-spezifisch ist.

41	Kategorie Einladungsprozess	Titel: Disable Version Control Support ist Sitzungs-global	minor/leicht	2

Die Einstellung "Disable Version Control" kann nicht pro Projekt gewählt werden bei einer Einladung - sie gilt für alle Projekte pauschal. Dies ist besonders in Kombination mit dem TabControl für die eingehenden Projekte ungünstig
Heuristics: H4 Konsistenz und Standards, H10 Hilfe und Dokumentation
Trigger: HE
Persistenz: Permanently

Änderung: Änderungen im Wizard führen dazu, dass die Option nun projektspezifisch angeboten wird.

42	Kategorie Einladungsprozess	Titel: Eingabe von existierenden Projektnamen ist umständlich und fehlerträchtig	moderate/mittel	2

Es sollte nur eine Wahl aus einer festen Liste der existierenden Projekte angeboten werden. Es hat keine nennenswerten Vorteile hier ein Eingabefeld zu platzieren.
Heuristics: H5 Fehler vermeiden
Trigger: HE
Persistenz: Permanently

Änderung: Auswahl der Projekte erfolgt nun aus einem Dropdown ohne freie Texteingabe.

43	Kategorie Einladungsprozess	Titel: Fehlende Größenangabe bei Einladung	minor/leicht	2

Saros gibt keine Information darüber wie groß die Projekte sind die verschickt werden. Ungünstig wenn nur noch wenig Festplattenspeichern verfügbar ist.
Heuristics: H1 Sichtbarkeit des Systemstatus, H5 Fehler vermeiden
Trigger: HE
Persistenz: Permanently

Änderung: Angabe der Größe und Dateianzahl wird nun übertragen beim Einladungsprozess

44	Kategorie Einladungsprozess	Titel: Lokales Farbschema fremdbestimmt	minor/leicht	2

Nutzer können für sich selbst eine Farbe wählen, mit der dann die Textänderungen und Selektionen hervorgehoben werden, sowie die Markierung der Viewports in Dateien. Somit hat der Nutzer keine Möglichkeit eine Farbe eines Sitzungsteilnehmers für sich lokal zu ändern, wenn er diese aufgrund von Systemeinstellungen nicht gut lesen kann, oder sie ihm subjektiv nicht gefällt
Heuristics: H3 Benutzerkontrolle und Freiheit, H4 Konsistenz und Standards
Trigger: UT [Solovjev2011] Nutzer wundert sich wieso er nur eigene Farbe ändern kann

Lösungsansatz: Änderung würde zu Problemen führen, da Nutzer verbalisieren "die rote Markierung" und nicht "meine Markierung". Hier wäre eine Verwirrung durch abweichende Farben problematisch.

45	Kategorie Einladungsprozess	Titel: Keine Funktion zum Rückgängigmachen von Selektionen bei Ressourcenauswahl	moderate/mittel	2

Es gibt kein Undo/Redo für Selektion von Ressourcen im Einladungsprozess. Verklickt der Nutzer sich, so muss er die Selektion erneut komplett vornehmen.
Heuristics: H5 Fehler vermeiden, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern
Trigger: HE
Persistenz: Repeatedly

Änderung: Undo/Redo Schaltflächen integriert

46	Kategorie Einladungsprozess	Titel: Non-Host hat kein Feedback über Fortschritt des Einladungsprozesses anderer Teilnehmer	minor/leicht	2

Sitzungsteilnehmer die nicht der Host sind, wissen nichts über den Fortschritt des Einladungsprozesses von anderen Nutzern. Saros zeigt lediglich einen Text "joining" an.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: HE

Änderung: InvitationStateCollector implementiert, der die Daten aus den Progress-Monitoren beim Host sammelt.

Lösungsansatz: Daten aus dem InvitationStateCollector könnten in den AwarenessInformationTreeElements dargestellt werden. Zu prüfen wäre hier das Update-Intervall, um den Einladungsprozess durch Informationsübermittlung nicht zu verlangsamen. Auch ist die Notwendigkeit einer derartigen Anpassung noch unklar.

47	Kategorie Einladungsprozess	Titel: Host hat keine Kontrolle über SVN-Verwendung	minor/leicht	2

Der Host kann nicht pro Sitzung, sondern nur global für Saros festlegen, dass SVN-Verbindungen genutzt werden können für die initiale Synchronisation der Projekte mit einer SVN-Bindung. Wenn der Host weiß, das die SVN-Anbindung für ein Projekt in der Einladung deutlich langsamer ist als ein direkter Transfer, so kann er dies nicht bequem deaktivieren, und es kann dazu kommen, dass der eingeladene Nutzer die SVN-Bindung nutzt.
Heuristics: H3 Benutzerkontrolle und Freiheit
Trigger: HE

Lösungsansatz: Host sollte pro Einladungsprozess wählen können ob SVN-Bindungen genutzt werden oder nicht.

48	Kategorie Einladungsprozess	Titel: Schreib/Leserechte nicht projektbezogen	moderate/mittel	2

Die Schreibrechte können nur pro Sitzung pro Nutzer vergeben werden durch den Host. Es ist keine Feingranulare Konfiguration möglich. Nutzer können nicht festlegen, dass bestimmte Projekte nur durch sie selbst schreibbar sind.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H3 Benutzerkontrolle und Freiheit
Trigger: HE

Lösungsansatz: Bessere Dokumentation des Rechtesystems anlegen, ggf. feingranularere Rechteverwaltung. Alternativ Rechteverwaltung gänzlich entfernen.

49	Kategorie Einladungsprozess	Titel: "Always run in background" wirkungslos	moderate/mittel	2

Die Einstellung "Always run in background" ist bei Saros-Fortschrittsbalken wirkungslos. Nutzer müssen Fortschrittsbalken stets manuell in den Hintergrund schicken, wenn sie dies wollen.
Heuristics: H3 Benutzerkontrolle und Freiheit
Trigger: HE + UT[Solovjev2011] Nutzer ist genervt bei immer wieder erscheinenden Fortschrittsbalken auf "Run in Background" klicken zu müssen
Typ: Nutzer äußert Frustration (verbal)

Änderung: Job-Familie eingeführt, sodass die Job-Fortschrittsbalken nun sowohl gruppiert werden, als auch eindeutig durch Eclipse identifiziert.

50	Kategorie Einladungsprozess	Titel: Kein automatisches Beenden einer leeren Sitzung	moderate/mittel	2

Host bleibt in leerer Sitzung nach dem letzter Teilnehmer die Sitzung verlässt.
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: UT - Der Host einer ersten Sitzung verblieb alleine in dieser Sitzung und hat dies zunächst nicht bemerkt. Er konnte dann nicht von anderen Probanden eingeladen werden, da er sich noch in einer Sitzung befand.
Qualifier: Missing (M)

Änderung: Dialog der den Host fragt, ob er die Sitzung beenden möchte, da er jetzt alleine ist. Nutzer kann diese Einstellung dauerhaft speichern.

51	Kategorie Einladungsprozess	Titel: Projekt überschreiben Dialog ohne Kopierfunktion	moderate/mittel	2

Entscheidet sich der Anwender, ein Projekt zu überschreiben, wird er eindringlich vor den möglichen Folgen eines Datenverlustes gewarnt. Der Dialog gibt keine Möglichkeit, eine Sicherheitskopie anzufertigen. Der Anwender ist gezwungen, den Dialog zu verlassen um eine Sicherheitskopie anfertigen zu lassen.
Heuristics: H3 Benutzerkontrolle und Freiheit, H5 Fehler vermeiden
Trigger: UT[Kahlert2011]

Lösungsansatz: Da Nutzer durch die Restrukturierung des Wizards nun wesentlich früher und deutlich verständlicher formuliert eine Option haben, um eine Kopie eines lokalen Projekts anzulegen, wird die Gewichtung des Problems verringert. Zudem ist das Problem durch den Nutzer leicht zu lösen, da dieser den Vorgang abbrechen kann und beim nächsten Versuch sofort die Kopier-Option nutzt. Die Frequenz des Auftretens ist als gering zu erwarten und die Persistenz ist sehr gering.

52	Kategorie Einladungsprozess	Titel: Datei-Annotationen werden nicht zuverlässig dargestellt	moderate/mittel	2

Durch Probleme mit seltenen Aktualisierungen der Darstellung von Navigator-Views in Eclipse fehlen teilweise die blauen Doppelpfeil-Icon-Markierungen, die Aufzeigen, dass eine Resource Bestandteil der Sitzung ist.
Heuristics: H1 Sichtbarkeit des Systemstatus, H6 Erkennen vor Erinnern
Trigger: HE
Typ: Eingeschränkte Effektivität
Qualifier: Missing (M)

Lösungsansatz: Prüfen der Implementierung für die Decoration-Klassen und ggf. forcieren einer Aktualisierung in festen Intervallen oder nach bestimmten Aktionen.

53	Kategorie Einladungsprozess	Titel: Offene Dateien werden nicht zu erst übertragen	minor/leicht	2

Die gerade geöffneten Dateien, die auch Bestandteil des Einladungsprozesses sind, sollten zuerst übertragen werden, damit ein Code-Review schneller starten kann.
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: UT [Solovjev2011] Nutzer erklärt offene Datei, obwohl die Übertragung noch läuft. Übertragung wurde via "run in background" ausgeblendet. Frustration, weil nun gewartet werden muss.
Typ: Nutzer äußert Frustration (verbal)
Persistenz: Repeatedly

54	Kategorie Einladungsprozess	Titel: Übertragungsende nicht bemerkbar bei "Run in background"	minor/leicht	2

Saros gibt keine Information darüber, ob die Einladung und Projektsynchronisation erfolgreich abgeschlossen wurde.
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: UT [Solovjev2011] Nutzer erklärt offene Datei, obwohl die Übertragung noch läuft. Übertragung wurde via "run in background" ausgeblendet. Frustration, weil nun gewartet werden muss. Eigene UTs: Das Beitreten eines Sitzungsteilnehmers wird von einigen Nutzern fehlinterpretiert als "Ende" des Prozesses.
Typ: Nutzer äußert Frustration (verbal)
Persistenz: Once

Änderung: Es ist durch die neue AwarenessStateCollector Klasse möglich, auch über den Einladungsprozess Informationen im Saros-View einzublenden. Hierzu existiert eine Klasse InvitationStateCollector. Allerdings ist dies nur als Prototyp implementiert worden, da der Bedarf sich als zu gering herausstellte um diesen Lösungsansatz weiter zu verfolgen.

Lösungsansatz: Ballon Benachrichtigung für erfolgreichen Abschluss der Übertragung

55	Kategorie Einladungsprozess	Titel: Fehlermeldung statt Info-Dialog bei Abbruch der Projektsynchronisation	minor/leicht	2

Es erscheint eine kryptische Fehlermeldung wenn der Nutzer die Projektsynchronisation abbricht.
Heuristics: H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern
Trigger: HE + UT[Kahlert2011] Sess-16

Änderung: Kryptische Fehlermeldung korrigiert.

Lösungsansatz: Abbruch der Projektsynchronisation ist gravierend, da so die Sitzung nicht aufgebaut werden kann, damit eigentlich akzeptabel das eine Fehlermeldung erscheint

56	Kategorie Einladungsprozess	Titel: Technische Informationen im Kopf des Wizards für eingehende Einladungen	minor/leicht	2

Im Kopfbereich des Wizards für ankommende Projekte wird beschrieben welche technische Verbindungsart genutzt wird und gewarnt, dass diese möglicherweise sehr langsam ist (IBB)
Heuristics: H10 Hilfe und Dokumentation
Trigger: HE - Eclipse Guideline + UT[Kahlert2011]

Änderung: Verschoben aus dem Kopf des Wizards in eine Info-Box auf dem Wizarddialog da es sich nicht um eine Validerungsmeldung handelt (Verstoß gegen Eclipse Guidelines)

Lösungsansatz: Untechnische Dokumentation bereitstellen, welche Verbindungsarten existieren und wie damit umgegangen werden soll

57	Kategorie Einladungsprozess	Titel: Hilfe für VCS-Deaktivierung abweichend von Styleguide	moderate/mittel	1

Die Hilfe ist nicht konform mit Eclipse-Styleguide und der Hinweistext ist fast unlesbar (grau auf grau).
Heuristics: H4 Konsistenz und Standards
Trigger: HE

Änderung: Hinweistext wurde entfernt, Integration der Beschreibung des Nutzens in den Einladungsprozess-Wizard.

58	Kategorie Einladungsprozess	Titel: Browse-Schaltfläche für Wahl von existierenden Projekten biete auch minor/leicht Ordner an	1

Dies suggeriert, dass man einen Ordner als Ziel wählen kann, auch wenn dann tatsächlich das Projekt verwendet wird. Selektion von Ordnern ist irreführend und überflüssig.
Heuristics: H4 Konsistenz und Standards, H8 Ästhetisches und minimalistisches Design
Trigger: HE
Persistenz: Permanently

Änderung: Durch Änderung des Wizards auf Listenselektion von existierenden Projekten entfällt dieses Problem

59	Kategorie Einrichtung	Titel: Wizards/Dialoge abgeschnitten dargestellt	severe/schwer	4

Tritt auf wenn Windows-Barrierefrei-Option 120% Elementgröße gewählt ist: Wizards werden ohne Scrollbalken und sonstige Hinweise auf Probleme mit der Fenstergröße abgeschnitten dargestellt. Nutzer kann nicht erkennen, dass Wizard zu klein dargestellt wird, da kein Element "partiell" dargestellt wird. Einrichtung kann nicht abgeschlossen werden.
Heuristics: H5 Fehler vermeiden
Trigger: UT - Tritt auf bei speziellen Windows-System-Einstellungen
Typ: Nutzer fragte nach Hilfe

Lösungsansatz: Wizards nicht mehr mit fixen Größen. Bzw. Wizards vergrößern oder Scrollbalken einführen.

60	Kategorie Einrichtung	Titel: Text im Getting Started Wizard schwer lesbar	severe/schwer	4

Durch prozentuale Größenberechnung der Wizard-Größe für den Getting Started Wizard und der automatischen verkleinerten Darstellung der Grafiken in dem Wizard sind diese schwer lesbar.
Heuristics: H4 Konsistenz und Standards
Trigger: Nutzer beschwerten sich, das sie den Text in den Bildern nicht entziffern können.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)
Persistenz: Permanently

Änderung: Der Getting Started Wizard wurde vollständig entfernt. Stattdessen wird eine Webseite in einem Browserfenster, in Eclipse intern, aufgerufen und stellt einen Teil der Saros-Webseite mit den Informationen aus dem bisherigen Wizard dar.

61	Kategorie Einrichtung	Titel: "Getting started" Wizard ist kein Wizard	severe/schwer	3

Das Saros-Tutorial ist als Wizard implementiert, was gegen mehrere Eclipse Richtlinien verstößt. Zudem ist der Wizard modal, und öffnet sich ungefragt, direkt nach der Installation von Saros.
Heuristics: H4 Konsistenz und Standards, H10 Hilfe und Dokumentation
Trigger: Eclipse Guideline, UT Nutzer versuchen mit den Screenshots zu interagieren, da sie nicht verstehen wieso in einem mehrseitigen Wizard nur statische Bilder dargestellt werden. Der Wizard fragt bei "Abbruch" (noch nerviger für Anwender) noch nach ob er beim nächsten Mal erneut angezeigt werden soll. Durch ungeschickte Formulierung der Frage neigen Nutzer dazu den Dialog mit Ja zu bestätigen, obwohl sie tatsächlich nicht wollten dass der Wizard noch mal angezeigt wird. Nutzer die das Tutorial gelesen haben, sind nicht schneller als die die es nicht gelesen haben. Nutzer übersprungen das Tutorial einfach, da keine Interaktion notwendig.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Better way (B)

Änderung: Das Tutorial wird nun in einem Eclipse-internen Web-Browser von der Webseite aus abgerufen. Damit ist es in direkter Nähe zum tatsächlich bedienbaren Saros-View, suggeriert weniger eine Interaktionsmöglichkeit die nicht gegeben wäre, und öffnet sich vor allem nicht modal. Der Nutzer kann sich im Tutorial informieren, während die Änderungen schrittweise in Saros nachvollzogen werden. Es stehen nun alle Möglichkeiten aus HTML,CSS und Javascript bereit für eine interaktive Führung. Es kann dort sogar eine Video-Einführung der Funktionen präsentiert werden.

Lösungsansatz: Das Tutorial kann noch deutlich interaktiver gestaltet werden. Beispielsweise kann das Tutorial konfigurationssensitiv werden, und dem Nutzer Tips anzeigne für den jeweiligen Einrichtungsschritt.

62	Kategorie Einrichtung	Titel: Whiteboard selten entdeckt & genutzt	moderate/mittel	3

Nutzer nehmen selten von der Existenz des Whiteboards Kenntnis und müssen explizit auf die Existenz des Saros Whiteboard hingewiesen werden, da es initial sehr unvorteilhaft als View geöffnet wird, und kein Zusammenhang zum Saros-View erkennbar ist. Dadurch das es keine leichte Möglichkeit gibt den View wieder herzustellen (kein Icon in der Menüleiste vom Saros-View, kein Eintrag im Saros-Menü)
Heuristics: H7 Flexibilität und effiziente Nutzung, H8 Ästhetisches und minimalistisches Design, H10 Hilfe und Dokumentation
Trigger: UT - Nutzer finden Whiteboard nicht / wissen nicht von dessen Existenz

Lösungsansatz: Menüeintrag "Open Whiteboard" in das Saros-Menü und ein Icon in die Menüleiste zum Ein/Ausblenden das Whiteboard-Views.

63	Kategorie Einrichtung	Titel: Add Buddy: Dialogelemente uneindeutig	minor/leicht	2

Bezeichnung "Nickname" beim Hinzufügen eines Buddies ist missverständlich, könnte als Namensteil

der XMPP ID verstanden werden.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt
Trigger: UT - Die Beschriftung "Nickname" wurde vom Probanden anders interpretiert (nicht als "Anzeigename") sondern als der Namensteil vor dem @-Zeichen in der XMPP ID (bei "alice@jabber.org" hätte er dort "alice" eingetragen). Anwender wollte Namensteil und Domain-Teil der XMPP ID getrennt eingeben, hat aber bemert, dass das nicht so konzipiert war. Nickname erschloss sich ihm zunächst nicht.
Qualifier: Incongruent mental model (I)
Persistenz: Once

64	Kategorie Einrichtung	Titel: Buddies-Baum ist zugeklappt nach hinzufügen des ersten Buddies	minor/leicht	2

Der Buddies-Baum wird beim erstmaligen Hinzufügen eines Buddies nicht automatisch ausgeklappt.
Heuristics: H6 Erkennen vor Erinnern
Trigger: UT - Der Nutzer stutzte zunächst und überlegte ob es mglw. nicht geklappt hat den Buddy hinzuzufügen, allerdings fand er schnell die Möglichkeit die Baumstruktur auszuklappen
Typ: Eingeschränkte Effizienz
Persistenz: Once

Lösungsansatz: Buddyliste immer ausklappen lassen, auch bei erstmaliger Einrichtung

65	Kategorie Einrichtung	Titel: "Buddy" Konotation unerwünscht	minor/leicht	2

Buddy bedeutet übersetzt "Kumpel" - dies ist eine Bezeichnung die für (unliebsame) Arbeitskollegen unangebracht ist aus Nutzersicht. Die Bezeichnung wurde bei der Entwicklung der Komponente aus den Begriffen von XMPP übernommen. Dort heißt die vordefinierte erste Gruppe von Kontakte "Buddies".
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H4 Konsistenz und Standards
Trigger: HE

Lösungsansatz: Umbennen zu "Contact".

66	Kategorie Einrichtung	Titel: Verbindungsversuche zum XMPP Server nicht abbrechbar	minor/leicht	2

Solange Saros versucht eine Verbindung aufzubauen zum XMPP Server, gibt es für den Nutzer keine Möglichkeit diesen Prozess zu unterbrechen. Dies kann bis zu 30 Sekunden dauern (konfigurierter Timeout). Dies ist nur problematisch bei langsamen Verbindungen und wenn zudem der XMPP-Server nicht erreichbar ist.

Heuristics: H3 Benutzerkontrolle und Freiheit
Trigger: HE + [Kahlert2011]
Persistenz: Permanently

67	Kategorie Einrichtung	Titel: Keine automatische Verbindung nach hinzufügen eines neuen Accounts	minor/leicht	2

Wenn Nutzer einen neuen XMPP-Account hinzufügen, müssen sie sich danach manuell zu diesem verbinden. Neue Accounts werden nicht automatisch als aktive Accounts übernommen und es erfolgt kein automatischer Verbindungsversuch. Dies weicht von den Nutzererwartungen ab.
Heuristics: H4 Konsistenz und Standards, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern
Trigger: UT[Kahlert2011]
Persistenz: Once

Lösungsansatz: Autoconnect-Checkbox hinzufügen. Prototyp existiert, allerdings ist ein direkter Autoconnect zu bevorzugen - kein Konsens im Saros-Team bisher wegen unklaren Privacy-Concerns bei Autoconnect.

68	Kategorie Einrichtung	Titel: Keine Integration von Saros in bestehende Systemarchitekturen	minor/leicht	2

Saros bietet abgesehen von der Wiedererwendung eines bestehenden XMPP-Accounts keine Möglichkeit Kontakte zu importieren (z.B: aus Skype). Alle Personen mit denen zusammen gearbeitet werden soll, müssen manuell hinzugefügt werden.
Heuristics: H7 Flexibilität und effiziente Nutzung
Trigger: Feature-Wunsch in Interviews

Lösungsansatz: Import-Funktion

69	Kategorie Einrichtung	Titel: Kein Saros-spezifisches Löschen/Sortieren/Filtern von Buddies möglich	minor/leicht	2

Nutzer die den XMPP-Account der für Saros eingesetzt wird auch anderweitig verwenden, möchten im Saros-View bestimmte Kontakte nicht aufgeführt haben. Saros bietet hierzu keine Möglichkeit. Das Löschen eines Kontaktes löscht diesen auch automatisch auf dem Server aus dem XMPP-Account (somit verschwinden Kontakte aus dem XMPP Account)
Heuristics: H3 Benutzerkontrolle und Freiheit, H5 Fehler vermeiden
Trigger: HE

Lösungsansatz: Trennung von XMPP-Account und Saros-Sicht auf Buddyliste.

70	Kategorie Einrichtung	Titel: Penetrante Meldungen über den Saros-Server	moderate/mittel	2

In diversen Wizards zur Einrichtung von XMPP-Accounts werden Informationstexte zum Saros-Server eingeblendet, auch wenn der Nutzer diesen nicht Nutzen möchte, und bereits explizit einen anderen Account konfiguriert hat.
Heuristics: H5 Fehler vermeiden, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern, H10 Hilfe und Dokumentation
Trigger: HE + UT - Nutzer lesen Anleitung lange und sind verwirrt ob es sie betrifft.

Lösungsansatz: Hinweise in eine Hilfe extrahieren. Erkennung einbauen ob der Nutzer mit dem Saros-Server verbunden ist, und ihn darauf hinweisen, das es nicht möglich ist mit Nutzern von XMPP-Accounts auf anderen Server eine Sitzung zu starten.

71	Kategorie Einrichtung	Titel: Dialogbox: "Request of subscription received" unklar	minor/leicht	2

Die Meldung für die Anfrage eines Kontakts, um diesen hinzuzufügen verwendet technische XMPP-Formulierungen die für den Nutzer völlig unverständlich sind.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H4 Konsistenz und Standards
Trigger: HE Nur einmal problematisch, da danach Text von Probanden nicht mehr gelesen wird, da Aktion bekannt. Wirkt aber unprofessionell.
Persistenz: Once

Änderung: Dialogtexte geändert

72	Kategorie Einrichtung	Titel: XMPP-Verbindungsversuch nicht abbrechbar	minor/leicht	2

Wenn Saros versucht eine Verbindung zum XMPP-Server aufzubauen, kann der Nutzer diesen Prozess nicht abbrechen. Es gibt keine Indikation wie lange der Prozess noch dauern kann.
Heuristics: H3 Benutzerkontrolle und Freiheit
Trigger: UT [Kahlert2011] HE - tritt fast nur bei langsamen Verbindungen auf
Persistenz: Permanently

73	Kategorie Einrichtung	Titel: Verbindung zum XMPP Server nicht im Saros-Menü	minor/leicht	2

Es gibt keinen Menüpunkt für die Verbindung zum XMPP Server im Saros-Menü
Heuristics: H4 Konsistenz und Standards
Trigger: UT[Kahlert2011] SaKo-4
Persistenz: Once

74	Kategorie Einrichtung	Titel: Verbindungszustand schwer zu erkennen	moderate/mittel	2

Es gibt keine eindeutige Indikation ob eine Verbindung zum XMPP Server besteht
Heuristics: H1 Sichtbarkeit des Systemstatus
Trigger: HE + UT + UT[Kahlert2011]

75	Kategorie Einrichtung	Titel: Probleme mit Einsatz bei Proxyservern	minor/leicht	1

Fehlermeldungen weil anonymisierter Upload von Nutzungsstatistiken nicht funktionieren kann. Error Log View erscheint ständig und lenkt ab.
Heuristics: H1 Sichtbarkeit des Systemstatus, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern
Trigger: HE+UT
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)

Lösungsansatz: Automatisches deaktivieren nach Fehlern bzw. RememberDecisionDialog mit Hinweis auf das Problem

76	Kategorie Einrichtung	Titel: Irreführende Meldung beim Editieren eines XMPP Accounts	minor/leicht	1

Edit XMPP/Jabber Account: "Your XMPP/Jabber ID must be in the format user@server.domain" ist inhaltlich falsch => "waldmann@awaldmann" ist genau so gültig.
Heuristics: H4 Konsistenz und Standards
Trigger: HE + UT - Nutzer lesen Beispiel und sind verwirrt

Lösungsansatz: Text anpassen und Beispiele für gültige XMPP-IDs geben.

77	Kategorie Einrichtung	Titel: [Kahlert2011] Benutzergruppen werden angezeigt sind aber nicht editierbar	minor/leicht	1

Die Benutzergruppen des verwendeten XMPP-Accounts werden angezeigt sind aber nicht editierbar.
Heuristics: H4 Konsistenz und Standards
Trigger: HE - UT [Kahlert2011]
Typ: Eingeschränkte Effektivität
Qualifier: Missing (M)

Lösungsansatz: Nutzergruppen editierbar machen.

78	Kategorie Einrichtung	Titel: Skype-Integration schwach dokumentiert	minor/leicht	1

Es ist nicht ausreichend dokumentiert, wozu die Skype-Integration dienen soll, und wie diese funktioniert.
Heuristics: H10 Hilfe und Dokumentation
Trigger: HE + UT [Kahlert2011] SaKo-Misc-2 Verwunderung über Skype-Integration und wie diese aussehen soll
Persistenz: Once

79	Kategorie Experimentelle Funktionen	Titel: Keine geeignete Indikation für aktive Voice-Over-IP Verbindung	severe/schwer	4

Es gibt keine geeignete Indikation dafür, dass eine Voice-over-IP Sitzung aktiv ist. Dies kann katastrophale Folgen haben, wenn Töne über das Mikrofon übertragen werden an andere Sitzungsteilnehmer, weil vergessen wurde die Verbindung zu beenden.
Heuristics: H1 Sichtbarkeit des Systemstatus, H3 Benutzerkontrolle und Freiheit, H4 Konsistenz und Standards, H5 Fehler vermeiden, H6 Erkennen vor Erinnern
Trigger: HE
Qualifier: Wrong (W)

80	Kategorie Experimentelle Funktionen	Titel: Keine Indikation das Screensharing aktiv ist	severe/schwer	4

Es gibt keine deutliche Anzeige, ob Screensharing aktiv ist
Heuristics: H1 Sichtbarkeit des Systemstatus, H4 Konsistenz und Standards
Trigger: HE

81	**Kategorie Experimentelle Funktionen**	**Titel: Nur Fremdkontrolle über geteilten Bereich des Screensharings**	**severe/schwer**	**4**

Der Nutzer der nur einen Ausschnitt seines Bildschirms teilen will (rund um den Mauszeiger), kann nicht verhindern, dass der andere Nutzer der zu einer Screensharing-Sitzung eingeladen wurde den Anzeigemodus ändert, und sich den gesamten Bildschirminhalt übertragen lässt
Heuristics: H3 Benutzerkontrolle und Freiheit, H4 Konsistenz und Standards

82	**Kategorie Experimentelle Funktionen**	**Titel: Screensharing Konfiguration ist extrem technisch**	**severe/schwer**	**4**

Die Einstellungen für das Screensharing sind detailliert möglich, dabei aber sehr technisch und erfordern Wissen über die Mechanismen des Screensharings, die ein durschnittlicher Nutzer nicht hat. Die erfolgreiche Einrichtung ist dem Laien damit so gut wie unmöglich. Es gibt keine geeignete Hilfe für die Einrichtung des Screensharings. Zudem muss ein Neustart erfolgen (damit Sitzungsabbruch), wenn Änderungen an der Konfiguration vom Screensharing erfolgen müssen.
Heuristics: H5 Fehler vermeiden, H10 Hilfe und Dokumentation
Trigger: UT[Solovjev2011] + HE Nutzer müssen Eclipse neu starten, nach Installation von Xuggler, um Screensharing mit Xuggler zu aktivieren (sonst erkennt Saros nicht das Xuggler installiert ist).

83	**Kategorie Experimentelle Funktionen**	**Titel: Screensharing: Fehlermeldungen sind nichtssagend und erscheinen beim falschen Nutzer**	**severe/schwer**	**4**

Diverse Fehlermeldungen durch defekte Konfiguration des Screensharings erscheinen nicht bei dem Nutzer der die Einladung zu einer Screensharing-Sitzung sendet, sondern bei dem eingeladenen Nutzer. Dies suggeriert dem Nutzer das er seine Konfiguration anpassen muss, auch wenn dies nicht der Fall ist. Nutzer versuchen ihre Einstellungen detailliert abzugleichen und vergeuden ihre Zeit damit.
Heuristics: H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern, H10 Hilfe und Dokumentation

84	**Kategorie Experimentelle Funktionen**	**Titel: Screensharing: Keine Indikation welcher Bereich des Bildschirms übertragen wird**	**severe/schwer**	**4**

Es gibt keine Anzeige, wie man es von anderen Tools gewöhnt sein könnte, die verdeutlicht welcher Teil des Bildschirms Bestandteil des Screensharings ist
Heuristics: H1 Sichtbarkeit des Systemstatus

85	Kategorie Experimentelle Funktionen	Titel: Experimentell: Voice-Over-IP Konfiguration sehr technisch	minor/leicht	2

Die Konfiguration der Voice-Over-IP Funktionalität ist sehr technisch. Es gibt keine Möglichkeit das Mikrofon auf Funktionalität zu testen (Vergleich Echo-Service in Skype)
Heuristics: H5 Fehler vermeiden
Trigger: HE

Lösungsansatz: Vereinfachen. Testfunktion einbauen. Ggf. Funktion entfernen, da unbrauchbar.

86	Kategorie Laufende Sitzung	Titel: Nicht-Texteditoren nicht synchronisiert	severe/schwer	4

Saros kann nur simple Texteditoren synchronisieren. Zusammengesetzte Editoren (z.B: für eine Plugin.xml oder build.properties, mit mehreren internen Tabs) können von Saros nicht synchronsiert werden. Es erscheint lediglich eine Bubble-Benachrichtigung, wenn man im Follow-Modus ist, das der Nutzer dem man folgt einen Editor geöffnet hat, der nicht übertragen werden kann.
Heuristics: H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern, H10 Hilfe und Dokumentation
Trigger: HE + UT - Ständige Verwirrung, was synchronisiert wird und was nicht

Lösungsansatz: Anzeige eines Views (Dummy-Editor) der auf die fehlende Synchronisationsmöglichkeit hinweist, da die Bubble-Benachrichtigung sich als zu subtil herausgestellt hat.

87	Kategorie Laufende Sitzung	Titel: Viewport-Indikation überlappend bei mehr als 2 Sitzungsteilnehmern	moderate/mittel	3

In einer Sitzung mit drei oder mehr Teilnehmern überlappen sich die Viewport-Markierungen in den Editor-Seitenleisten, sodass nicht mehr ersichtlich ist,
Heuristics: H6 Erkennen vor Erinnern, H8 Ästhetisches und minimalistisches Design
Trigger: Dies ist in Usability-Tests der Prototypen aufgefallen. Nutzer konnten sich anhand der Präsenzinformationen im Saros-View orientieren.
Typ: Eingeschränkte Effektivität
Qualifier: Better way (B)

Änderung: Durch die Veränderung des Saros-Views ist zumindest eindeutig, das der Nutzer sich in der Datei aufhält.

Lösungsansatz: Viewport-Markierungen verändern, ggf. spezifisch für Sitzungen mit mehr als 2 Nutzern, oder besser allgemein im Fall das 3 Nutzer sich denselben Bereich einer Datei ansehen die Markierungen anders darstellen.

88	**Kategorie Laufende Sitzung**	**Titel: Mangelnde Performance bei großen Datei-Veränderungen**	**moderate/mittel**	**3**

Wenn ein Nutzer in Eclipse Funktionen z.B. zum Autoformatieren nutzt (Strg+Shift+F) oder in Java Code Refactoring Operationen duchführt, erfolgt die Übertragung nicht in einem Rutsch sondern Zeichen für Zeichen. Bei einer langsamen Internetverbindung müssen Sitzungsteilnehmer somit sehr lange warten, bis das Ergebnis der Veränderungen bei ihnen nachvollzonge wurde. Der Prozess ist nicht abbrechbar, d.h. der Nutzer muss warten bis er abgeschlossen ist.
Heuristics: H3 Benutzerkontrolle und Freiheit, H7 Flexibilität und effiziente Nutzung
Trigger: UT [Solovjev2011] - Nutzer formatiert XML-Datei. Mehrere Minuten Wartezeit bis Formatierung erledigt.
Typ: Nutzer äußert Frustration (verbal)

Lösungsansatz: Komplexe Operationen abfangen und zusammenfassen in ein Datenpaket das an die Sitzungsteilnehmer versandt wird. Erste Ansätze gibt es im Code bereits, allerdings ist unklar ob diese funtionsfähig sind. Ggf. Prozess "abbrechbar" machen und per Watchdog in einem Rutsch alle inkonsistenten Dateien nachladen.

89	**Kategorie Laufende Sitzung**	**Titel: Fehlende Übertragung von Eclipse-Inhalten**	**moderate/mittel**	**3**

Saros überträgt die Inhalte und Interaktionen diverser Views und Funktionen von Eclipse nicht, ohne dass dies dem Benutzer verdeutlicht wird bei der Nutzung: - Suchfunktion & Suchergebnisse - Konsolenausgaben - Debug - Tooltips (z.B. Javadocs)
Heuristics: H7 Flexibilität und effiziente Nutzung, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern, H10 Hilfe und Dokumentation
Trigger: UT [Solovjev2011] + eigene UT + HE
Typ: Eingeschränkte Effizienz
Persistenz: Permanently

90	**Kategorie Laufende Sitzung**	**Titel: Verwirrung über Synchronisierte Teile/nicht synchronisierte Teile**	**moderate/mittel**	**3**

Es ist undeutlich, welche Informationen Saros übermittelt und welche nicht. Zwar trat das Problem vor allem in einem Usability-Test auf, bei dem auch eine Videoaufzeichnung erfolgte (woraufhin Probanden dachten, das auch ihr Bildschirm übertragen wird), allerdings ist insgesamt aufällig, dass Saros nichts unternimmt um dem Nutzer visuell klar zu machen, wenn ein offener View übertragen wird oder nicht.

Heuristics: H6 Erkennen vor Erinnern
Trigger: UT [Solovjev2011] + HE Probanden sind verwirrt und fragen nach was der andere Nutzer sieht.
Typ: Eingeschränkte Effizienz
Qualifier: Incongruent mental model (I)
Persistenz: Repeatedly

Lösungsansatz: Andersfarbige Rahmenlinie / Overlays / sonstige Darstellungsänderungen für die Views die von Saros synchronisiert werden.

91	Kategorie Laufende Sitzung	Titel: Buddyliste redundant zu Sitzungsteilnehmer-Liste	moderate/mittel	3

Die Buddyliste bleibt bestehen in einer laufenden Sitzung und beinhaltet auch die Sitzungsteilnehmer. Damit existieren Baumeinträge "doppelt" mit unterschiedlichen Kontextmenüs
Heuristics: H4 Konsistenz und Standards
Trigger: HE+UT [Solovjev2011] Verwirrung bei Nutzer wie man den Followmodus aktiviert, da Einträge doppelt (in Sitzung+Roster) mit unterschiedlichen Kontextmenüs im selben Tree (sic!)
Typ: Eingeschränkte Effizienz
Persistenz: Repeatedly

Änderung: Filterung implementiert sodass Sitzungsteilnehmer nicht mehr in der Buddyliste aufgelistet werden.

92	Kategorie Laufende Sitzung	Titel: Farben werden nicht Sitzungsübergreifend beibehalten	moderate/mittel	3

Die Benutzer-Farben werden nicht Sitzungsübergreifend beibehalten (auch nicht wenn ein Nutzer seine Farbe explizit wechselt).
Heuristics: H4 Konsistenz und Standards
Trigger: Eigene UTs + UT [Kahlert2011] Sess-9 Nutzer wundern sich wieso sie in einer weiteren Sitzung plötzlich eine neue Farbe haben.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Wrong (W)
Persistenz: Permanently

Lösungsansatz: Speichern der Farbkonstellationen für Nutzerkombinationen in Sitzungen. Aufgrund des Colormanagements schwierig.

93	Kategorie Laufende Sitzung	Titel: Readonly entspricht nicht den Erwartungen	severe/schwer	3

Als Readonly Nutzer kann man über Eclipse-Funktionen Refactorings durchführen. Auch wird die Tastenkombination Strg+D zum löschen der aktuellen Zeile nicht abgefangen. Es ist möglich die Änderungen zu speichern. Zudem kann ein READONLY Nutzer mit Drag & Drop Operationen Änderungen herbeiführen.

Heuristics: H4 Konsistenz und Standards, H10 Hilfe und Dokumentation

Trigger: HE + [Kahlert2011] - Rght-6

Qualifier: Wrong (W)

Persistenz: Permanently

Lösungsansatz: Ein Bugfix ist in Arbeit - es liegt hier ein Software-Fehler vor. Da die Funktion sehr selten genutzt wird eher unkritisch.

94	Kategorie Laufende Sitzung	Titel: Keine Information welche Dateien verändert wurden	moderate/mittel	2

Saros stellt keine Information darüber bereit, welche Dateien in einer Saros-Sitzung verändert wurden, und durch wen. Wenn einer der Entwickler den Code z.B. in ein SVN Repository commited, fehlt zunächst jegliche Information darüber, wer an den Änderungen beteiligt war.

Heuristics: H6 Erkennen vor Erinnern

Trigger: HE + Interviews

Änderung: Changelog implementiert, das protokolliert welcher Nutzer welche Datei in einer Sitzung geöffnet hat.

Lösungsansatz: Nutzer sollten das Changelog öffnen können über eine Schaltfläche. Es sollte sich nach Möglichkeit automatisch aktualisieren, und mehr Informationen bereitstellen. Auch wäre es sinnvoll, wenn das System beschreibt, welche Dateien sich tatsächlich inhaltlich geändert haben.

95	Kategorie Laufende Sitzung	Titel: Scrolling im Followmode kollidiert mit Scrolling des Drivers	minor/leicht	2

Solange man im Followmodus ist, wird die gerade offene Datei automatisch mitgescrollt sobald der Nutzer, dem man folgt, anfängt zu scrollen. Scrollt man nun selbst, so kollidieren die Scrollaktivitäten und Saros scrollt erneut zum sichtbaren Teil des Editors von dem Nutzer dem gefolgt wird. Weder wird der Followmodus durch die höher zu gewichtende Nutzerinteraktion automatisch verlassen, noch hat der Nutzer eine Möglichkeit ihn temporär zu pausieren. Der Nutzer muss den Followmodus manuell reaktivieren wenn er beendet wurde, über Auswahl des Zielnutzers und Nutzen des Menüleisten-Icons oder des Kontextmenüs

Heuristics: H4 Konsistenz und Standards, H10 Hilfe und Dokumentation
Trigger: UT - Nutzer kommentiert das seitliches Scrolling nicht klappt. Nutzer fügen manuell Zeilenumbrüche ein aus Rücksichtsnahme.
Qualifier: Incongruent mental model (I)
Persistenz: Permanently

Lösungsansatz: Dokumentation hinzufügen das seitliches Scrolling nicht synchronisiert wird.

96	Kategorie Laufende Sitzung	Titel: Eigenständiges Scrolling im Follow-Modus nicht möglich	minor/leicht	2

Es bestand der Wunsch bei Probanden, separat in der Datei zu scrollen, um so Inhalte betrachten zu können, die der Driver vor kurzem erläutert hatte. Observer die in einem Code-Review den Code langsamer verstehen als der Driver erklärt, scheuen sich offenbar dem Driver dies mitzuteilen (XXX). Dieses Problem enstand vermutlich daraus, dass es keine expliziten Driver-/Observer-Rollen mehr gibt, und damit der Follow-Modus als technischer Ersatz extrahiert wurde.
Heuristics: H3 Benutzerkontrolle und Freiheit
Trigger: UT - Nutzer äußerten Feature-Wunsch

97	Kategorie Laufende Sitzung	Titel: Host-Rolle nicht übertragbar	moderate/mittel	2

Die Host-Rolle ist nicht übertragbar. Will ein Host die Sitzung verlassen, können verbleibende Sitzungsteilnehmer nicht weiterarbeiten, sondern müssen erst eine eigene Sitzung starten.
Heuristics: H3 Benutzerkontrolle und Freiheit, H7 Flexibilität und effiziente Nutzung
Trigger: UT - Zeitverschwendung durch nötigen Sitzungsneustart. Driver/Observer-Tausch zudem nicht realisierbar
Typ: Eingeschränkte Effizienz
Persistenz: Permanently

Lösungsansatz: Host-Rolle übertragbar machen.

98	Kategorie Laufende Sitzung	Titel: Fehlermeldung für den Host bei Ablehnen einer Einladung	minor/leicht	2

Lehnt ein Nutzer eine Einladung ab, erscheint beim Host ein Fehlerdialog zusätzlich zu einer Balloon Benachrichtigung.
Heuristics: H4 Konsistenz und Standards

Trigger: Nutzer wird verunsichert durch "agressive" Fehlermeldung.
Qualifier: Better way (B)
Persistenz: Permanently

Änderung: Balloon-Notification wurde deaktiviert, da Redundant.

Lösungsansatz: Error-Dialog ersetzen durch Message-Dialog (leider nicht-triviale Änderung aufgrund der Codestruktur)

99	Kategorie Laufende Sitzung	Titel: [Kahlert2011] Verwirrung darüber, dass nur der Host die Rechte ändern kann.	minor/leicht	2

Nutzern erschließt sich nicht schnell, das nur der Host die Rechte in einer Sitzung ändern kann.
Heuristics: H10 Hilfe und Dokumentation
Trigger: UT [Kahlert2011]
Qualifier: Incongruent mental model (I)

Lösungsansatz: Bessere Dokumentation der Rolle im Saros-Tutorial

100	Kategorie Laufende Sitzung	Titel: Kann Nutzer aus Sitzung nicht entfernen	minor/leicht	2

Es gibt keine Möglichkeit, Nutzer aus einer laufenden Sitzung zu entfernen. Diese müssen eigenständig die Sitzung verlassen.
Heuristics: H4 Konsistenz und Standards
Trigger: HE
Persistenz: Permanently

Lösungsansatz: Der Host sollte Nutzer aus der Sitzung entfernen können.

101	Kategorie Laufende Sitzung	Titel: Kann Projekte/Dateien nicht aus Sitzung entfernen	moderate/mittel	2

Man kann Projekte/Dateien nicht aus einer laufenden Sitzung entfernen (unshare)
Heuristics: H4 Konsistenz und Standards
Trigger: HE
Persistenz: Permanently

Lösungsansatz: Implementierung einer Unshare-Funktionalität, die die Datei aus der Sitzung entfernt. Dies ist bei "Full Sharing" von Projekten schwierig.

102	Kategorie Laufende Sitzung	Titel: Verwirrender Speichern-Dialog wenn Host geänderte Datei ungespeichert schließt	moderate/mittel	2

Nutzer im Followmodus zu einem anderen Nutzer sehen einen verwirrenden Speichern-Dialog, wenn der gefolgte Nutzer eine veränderte Datei ungespeichert schließt. Betrifft Änderungen von beiden Nutzern, die dabei verloren gehen wenn mit "Nein" nicht gespeichert wird - Inkonsistenz bei "Ja". Unklare Situation bei "Abbrechen"
Heuristics: H10 Hilfe und Dokumentation
Trigger: UT
Persistenz: Repeatedly

Lösungsansatz: Hierzu gab es endlose Diskussionen über das korrekte Verhalten im Review-Board bezüglich des Verhaltens bei ungespeicherten Änderungen in einem Editorpart. http://saros-build.imp.fu-berlin.de/reviews/r/176/

103	Kategorie Laufende Sitzung	Titel: Zweifache Viewport-Indikation unklar	minor/leicht	2

Selbst Eclipse-Veteranen ist die Viewport-Indikation LINKS und RECHTS des Editor-Views unklar (wozu die Doppelung).
Heuristics: H10 Hilfe und Dokumentation
Trigger: UT [Solovjev2011] - Nutzer fragen nach dem Viewport des anderen Nutzers, anstatt die seitlichen Farbbalken dafür zu nutzen
Qualifier: Overlooked (O)
Persistenz: Repeatedly

Lösungsansatz: Der Bereich Awareness-Informationen genügt als Thema für eine Master- oder Diplomarbeit. Es gibt bereits eine Arbeit, die sich um die Verbesserung kümmert, die im Anschluss an diese Diplomarbeit bearbeitet wird.

104	Kategorie Laufende Sitzung	Titel: Partial Sharing bei Library-Abhängigkeiten problematisch	minor/leicht	2

Es gibt keinen Hinweis (obwohl das eigentlich klar sein sollte) darauf, dass beim Partial Sharing mögliche Abhängigkeiten nicht aufgelöst werden, und somit beispielsweise Java-Code scher zu lesen ist, durch die unzähligen fehlenden referenzierten Klassen und Bibliotheken.
Heuristics: H5 Fehler vermeiden, H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern

Trigger: UT [Solovjev2011] - Proband beschwert sich, dass die Libraries nicht nachgeladen werden (im Hintergrund)
Typ: Nutzer äußert Frustration (verbal)
Persistenz: Repeatedly

Lösungsansatz: Bessere Dokumentation / Warnhinweise. Mglw. Biblibotheken (JARs) nachladen im Hintergrund?

105	**Kategorie Laufende Sitzung**	**Titel: Kann nicht kompilieren bei parallelem Arbeit und Fehlern bei anderen Sitzungsteilnehmern**	**minor/leicht**	**2**

Wenn zwei Nutzer parallel in einem Projekt arbeiten, und Nutzer A in seinem Codeteil Fehler hat, schlägt auch bei Nutzer B das Kompilieren fehl, da die Änderungen permanent synchronisiert werden.
Heuristics: H10 Hilfe und Dokumentation
Trigger: UT [Solovjev2011] - Nutzer ist verärgert, dass er nicht kompilieren kann, weil der andere Nutzer an anderer Stelle etwas modifiziert hatte.
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Incongruent mental model (I)
Persistenz: Once

Lösungsansatz: Dies ist primär ein Problem im mentalen Model der Nutzer, die parallel am selben Projekt arbeiten wollen, aber ohne die negativen Aspekte durch Programmfehler, die kollaborativ entstanden sind. Allerdings sollten derartige Eigenschaften von Saros-Sitzungen besser dokumentiert sein.

106	**Kategorie Laufende Sitzung**	**Titel: Technische Begriffe für Verbindungstyp an Buddy**	**minor/leicht**	**2**

Es erscheinen technische Informationen am Buddyelement im Saros-View, wenn eine Verbindung besteht, z.B. "connected via Socks 5 (mediated)"
Heuristics: H2 Übereinstimmung zwischen System und realer Welt
Trigger: HE - Nutzerumfragen
Persistenz: Permanently

Lösungsansatz: Hilfe und Dokumentation muss angepasst werden, um den Nutzer darauf hinzuweisen, wie er die Verbindungsqualität verbessern kann.

107	**Kategorie Laufende Sitzung**	**Titel: Keinerlei Feedback beim Senden einer Datei**	**moderate/mittel**	**2**

Es gibt beim Senden einer einzelnen Datei über das Kontextmenü eines Situngsteilnehmers kein geeignetes Feedback über den Fortschritt/Erfolg. Der Job ist als SystemJob implementiert und wird damit dem Nutzer nur im "Progress View" von Eclipse dargestellt. Für die meisten Nutzer ist dies somit weitestgehend unsichtbar.
Heuristics: H1 Sichtbarkeit des Systemstatus, H3 Benutzerkontrolle und Freiheit
Trigger: UT [Solovjev2011]: Frustration wegen fehlender Rückmeldung des Systems.
Typ: Nutzer äußert Frustration (verbal)
Persistenz: Permanently

Änderung: Fortschrittsbalken werden dargestellt, da der Job auf "User"-Job umgestellt wurde. Nutzer können auch den Transfer abbrechen.

108	Kategorie Laufende Sitzung	Titel: Technische Anzeige des Verbindungstyps	minor/leicht	2

An den Buddies wird angezeigt, welche technische Verbindungsart eingesetzt wird
Heuristics: H10 Hilfe und Dokumentation
Trigger: HE (+ UT: Nutzer vermuten, dass die Sitzung startet weil ein Hinweis auf die Verbindungsart am Buddy im Saros-View erscheint)

Änderung: Diese Textinformationen entfallen durch die Filterung der Buddyliste, sodass Sitzungsteilnehmer nicht mehr doppelt aufgelistet werden

Lösungsansatz: Untechnische Dokumentation bereitstellen, welche Verbindungsarten existieren und wie damit umgegangen werden soll

109	Kategorie Laufende Sitzung	Titel: Host kann sich selbst Schreibrechte nicht entziehen	minor/leicht	1

Auch wenn dies zunächst komisch wirkt: der Host kann sich die Schreibrechte nicht entziehen. Da die Host-Rolle nicht übertragbar ist, ist klassische Paarprogrammierung mit festen Driver-/Observer-Rollen nicht möglich, da der Host immer schreiben kann. Zwar ist dies eine der Hauptfunktionen von Saros, das alle Teilnehmer zunächst gleichberechtigt starten, was auch positiv aufgenommen wird von den Nutzern, allerdings wunderten sich einige Probanden über die fehlende Möglichkeit in einer 2er-Sitzung die "Rollen" zu tauschen (was ja durch Änderung der Schreibrechte erfolgen würde)
Heuristics: H4 Konsistenz und Standards
Trigger: UT/I
Qualifier: Incongruent mental model (I)

Lösungsansatz: Die Host-Rolle sollte übertragbar werden.

110	Kategorie Laufende Sitzung	Titel: Baumdarstellungen für Sitzungsteilnehmer speichert Zustand nicht	minor/leicht	1

Die Baumdarstellung für die Sitzungsteilnehmer wird von Saros automatisch ausgeklappt, sobald eine Aktivität eines Nutzers erfolgt ist.
Heuristics: H4 Konsistenz und Standards
Trigger: Eclipse-Guideline legt fest, das Views ihren Zustand speichern müssen. Zudem erwarten Nutzer, dass das Einklappen einer Baumstruktur (wenn es möglich ist) permanent erfolgt.
Typ: Eingeschränkte Effektivität
Qualifier: Incongruent mental model (I)

Lösungsansatz: Dies ist "a Feature not a bug" und hat sich im Praxiseinsatz schon bewährt. Nutzer klappen zwar gerne die Baumdarstellung zu, versäumen aber, diese wieder auszuklappen, um die hinterlegten Informationen einzublenden. Das automatische wieder Ausklappen der Baumstruktur wurde nicht als negativ wahrgenommen.

111	Kategorie Laufende Sitzung	Titel: Kollaboratives Strg+Z unklar	minor/leicht	1

Es ist unklar, ob Strg+Z nur lokale Änderungen rückgängig macht, oder auch die von anderen Sitzungsteilnehmern
Heuristics: H10 Hilfe und Dokumentation
Trigger: UT [Solovjev2011] Nutzern war unklar, was passiert, wenn sie Strg+Z drücken oder die "Rückgängig"-Aktion auslösen. Einmaliges Probieren reichte zur Erkenntnis.
Qualifier: Incongruent mental model (I)
Persistenz: Once

112	Kategorie Preferences	Titel: Preferences: Latency Measurement unklar	minor/leicht	2

Für die Einstellung "Performance latency measurement using Ping Pong activities" ist völlig unklar, zu welchem Zweck diese eingeschaltet werden kann
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H10 Hilfe und Dokumentation
Trigger: HE

Lösungsansatz: Unklare Bezeichner ersetzen, Hilfe bereitstellen.

113	Kategorie Preferences	Titel: Preferences: Start sessions in follow mode	minor/leicht	2

Die Funktion "start sessions in follow mode" ist uneindeutig für Sitzungen mit mehr als 2 Nutzern. Der

Nutzer kann nicht wissen, welchem Nutzer er initial folgen wird
Heuristics: H4 Konsistenz und Standards, H10 Hilfe und Dokumentation
Trigger: HE

114	Kategorie Preferences	Titel: Preferences: "Use needs-based synchronization"	minor/leicht	2

Völlig unklare Funktionalität, die nicht erklärt wird.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H10 Hilfe und Dokumentation
Trigger: HE

115	Kategorie Preferences	Titel: Preferences Communication: "Beep when receiving a chat message" falsch	minor/leicht	2

Beept auch bei Online/Offline-Statusänderung eines Buddies. Sowie bei Empfang einer Nachricht. Redundant zum Toolbar-Icon und uneindeutig.
Heuristics: H4 Konsistenz und Standards
Trigger: HE

116	Kategorie Preferences	Titel: Preferences: uneindeutige Formulierung "Enable concurrent undo (only local changes are undone, session restart necessary)"	minor/leicht	1

Suggeriert, dass ein Sitzungsneustart für ein Undo notwendig wäre. Zudem ist dem Nutzer völlig unklar, welche Vorteile er durch diese Option hätte. Es ist eindeutig eine "advanced" option. Kein Saros-Entwickler konnte sagen, was die Funktion tut.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt
Trigger: HE

117	Kategorie Preferences	Titel: Preferences: Do not use version control support	minor/leicht	1

Negative Formulierung der Funktionalität ist nicht konsistent mit anderen Einstellungen. Vorteile/Nachteile der Funktionalität völlig unklar
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H4 Konsistenz und Standards, H10 Hilfe und Dokumentation
Trigger: HE

118	Kategorie Preferences	Titel: Preferences Communication	minor/leicht	1

Formatierung der Groups entspricht nicht Eclipse Styleguide. Konfiguration des Skype-Namens ist dem Nutzer völlig unklar.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H4 Konsistenz und Standards, H8 Ästhetisches und minimalistisches Design, H10 Hilfe und Dokumentation
Trigger: HE

119	Kategorie Saros-View	Titel: Balloon-Benachrichtigungs-Verhalten nicht Erwartungskonform	severe/schwer	4

Die Balloon-Benachrichtigungen verhalten sich nicht erwartungskonform: - Sie sind nur über das kleine Kreuzchen schließbar, der Rest der Balloon-Notification ist nicht zum Schließen geeignet. - Sie öffnen sich an merkwürdigen Stellen, wenn der Saros-View minimiert oder geschlosssen ist - Lange Meldungen ragen aus dem Bildschirm heraus (bei Mehr-Bildschirm-Konfigurationen)
Heuristics: H4 Konsistenz und Standards
Trigger: UT - Nutzer reagieren emotional und vor allem negativ, dass ein Popup einen Teil ihres Editors überdeckt und für sie uninteressante Informationen darstellt. Besonders regen sie sich darüber auf, das die Meldungen sich nur per Kreuz-Icon schließen lassen. Mehrfaches erscheinen überlappender Benachrichtigungen führt dazu das Nutzer Zeit vergeudet in dem er alle schließt
Typ: Nutzer äußert Frustration (verbal)
Qualifier: Better way (B)

Änderung: Anpassungen der Handhabung: Jeder Klick auf die Meldung schließt diese. Jede Anzeige einer Meldung löscht vorherige Meldungen, sodass nicht (z.B. beim Tippen als READONLY-Nutzer) mehrere Benachrichtigungen hintereinander erscheinen.

Lösungsansatz: Automatische Zeilenumbrüche für die Meldungen wäre noch sinnvoll. Noch besser wäre es, diese zu beseitigen und durch Meldungen, die innerhalb des Saros-Views erscheinen, zu ersetzen. Auch wäre es sinnvoll diese Meldungen deaktivieren zu können.

120	Kategorie Saros-View	Titel: Ungünstige Reihenfolge in Buddy-Kontextmenüs	moderate/mittel	3

Die Einträge in den Kontextmenüs für Buddies sind nicht sinnvoll angeordnet: Der am häufigsten genutzte Eintrag "Work together on..." findet sich erst am Ende des Kontextmenüs.
Heuristics: H4 Konsistenz und Standards, H7 Flexibilität und effiziente Nutzung
Trigger: Die am häufigsten genutzten Menüeinträge sollten ganz oben erscheinen. Dies vermeidet zudem "Mauskilometer".
Typ: Eingeschränkte Effizienz
Persistenz: Permanently

Änderung: Kontextmenü-Reihenfolge verändert.

121	Kategorie Saros-View	Titel: Farbfläche hinter Sitzungsteilnehmer in Baumdarstellung entspricht nicht Klickfläche	severe/schwer	2

Die Farbfläche hinter dem Baumeintrag für einen Sitzungsteilnehmer entspricht nicht der klickbaren Fläche mit der das Kontextmenü auf diesem Element ausgelöst werden kann. Lediglich ein Klick direkt auf den Texttteil des Baumeintrags (also den Namen des Buddies) öffnet das korrekte Kontextmenü.
Heuristics: H4 Konsistenz und Standards, H8 Ästhetisches und minimalistisches Design
Trigger: UT - Dadurch, dass sich immer ein Kontextmenü öffnet, egal wo man klickt erschließt sich Nutzern nicht sofort, dass sie auf den Namen klicken müssen. Dadurch bemerken Nutzer erst deutlich später Systemfunktionen, die über das Kontextmenü eines Sitzungsteilnehmers zugreifbar sind.
Qualifier: Overlooked (O)
Persistenz: Repeatedly

Änderung: Durch Veränderung des Saros-Views und Ersetzen der Farbfläche durch ein passend platziertes Farbquadrat entfällt dieses Problem

122	Kategorie Saros-View	Titel: Kein Start-Knopf	minor/leicht	2

Saros bietet keine Schaltfläche zum Starten des Einladungsprozesses mit nur einem Klick. Nutzer müssen explizit das Saros-Menü anwählen und dort einen Untermenüpunkt oder die Kontextmenüs nutzen.
Heuristics: H7 Flexibilität und effiziente Nutzung, H8 Ästhetisches und minimalistisches Design
Trigger: HE + UT - Nutzer suchen nach Möglichkeiten, eine Sitzung zu starten, finden nur die Schnellzugriffsvarianten, die keinen Wizard für die Einladung anzeigen, sondern die Enladung direkt starten.
Typ: Eingeschränkte Effektivität
Qualifier: Better way (B)

Änderung: Schaltfläche hinzugefügt zu Dashboard-Prototypen

123	Kategorie Saros-View	Titel: Chat nicht abdockbar (für größere Darstellung)	minor/leicht	2

Für Nutzer, für die der Chat wichtig ist, gibt es keine Möglichkeit den Chat-Teil aus dem Saros-View herauszulösen (abzudocken) und diesen anders anzuordnen.
Heuristics: H4 Konsistenz und Standards, H7 Flexibilität und effiziente Nutzung
Trigger: HE - Abweichung von Eclipse Styleguide

Lösungsansatz: Möglichkeit integrieren, den Chatview aus dem Saros-View zu entkoppeln und diesen separat anzuordnen.

124	Kategorie Saros-View	Titel: Icon für "Sitzung verlassen" identisch zu "Sitzung beenden"	minor/leicht	2

Das Icon für die Aktion "Sitzung verlassen" ist identisch zu dem Icon für "Sitzung beenden" wenn man der Host ist.
Heuristics: H4 Konsistenz und Standards
Trigger: UT - Nutzer "verlassen" die Sitzung nicht, da sie denken, dass sie die Sitzung für alle beenden würden. Der Host wird gebeten, die Sitzung zu beenden.
Qualifier: Incongruent mental model (I)
Persistenz: Repeatedly

Lösungsansatz: Icon austauschen. Es existiert bereits ein Icon mit einem Tür-Symbol, das allerdings wenig geeignet erscheint.

125	Kategorie Saros-View	Titel: Kryptische Fehlermeldung wenn XMPP Account auf anderem Computer genutzt wird	minor/leicht	2

Saros View zeigt "Error (stream:error(text))" wenn sich jemand mit dem verwendeten XMPP Account auf einem anderen Computer anmeldet (oder mit einer anderen Software).
Heuristics: H9 Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern
Trigger: HE

Änderung: Balloon-Nachricht wird eingeblendet die den Nutzer informiert, dass sein Account woanders genutzt wird.

126	Kategorie Saros-View	Titel: SarosView geht verloren bei Perspektivenwechsel	minor/leicht	2

In einigen Eclipse-Konfigurationen ist der Saros-View nach einem Perspektiven-Wechsel nicht mehr sichtbar (z.B. Debug-Perspektive)
Heuristics: H4 Konsistenz und Standards
Trigger: UT [Solovjev2011]
Typ: Eingeschränkte Effektivität
Persistenz: Once

Lösungsansatz: Saros sollte in allen Perspektiven einheitlich eingeblendet werden, aber nicht mehr dargestellt werden, wenn der Nutzer bewusst den Saros-View entfernt.

127	Kategorie Saros-View	Titel: Tab-Leiste für Chatraum suggeriert Möglichkeit mehrerer Chaträume	minor/leicht	1

Die Darstellung eines Tabcontrols für den einzigen Chatraum den Saros zurzeit unterstützt, suggeriert für den Nutzer, das dort mehrere Tabs offen sein können. Der Nutzer kann so versucht sein, herauszufinden, wie er weitere Chats öffnen kann, und vergeudet seine Zeit.
Heuristics: H4 Konsistenz und Standards, H8 Ästhetisches und minimalistisches Design
Trigger: HE

Änderung: Einführung eines Übersicht-Tabs bzw. eines Willkommen-Tabs mit Schaltfläche, der permanent vorhanden ist.

Lösungsansatz: Integration von Instant Messaging Funktionalität, also der Möglichkeit mit anderen Nutzern zu zweit chatten zu können ohne dafür in einer gemeinsamen Sitzung sein zu müssen.

128	Kategorie Saros-View	Titel: Chat: Bezeichnung "roundtable" unklar	minor/leicht	1

Die Bezeichnung "Roundtable" ist schlecht aus dem Deutschen ins Englische übersetzt und hat keine Bedeutung für den Nutzer.
Heuristics: H2 Übereinstimmung zwischen System und realer Welt, H4 Konsistenz und Standards
Trigger: HE

Änderung: Umbenannt zu "Session Chat"

8 Quellenverzeichnis

[Djemeili2006] R. Djemili. Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung verteilter Paarprogrammierung. Master's thesis, Freie Universität Berlin, Institut für Informatik, 2006

[Kahlert2011] Bjorn Kahlert. Verbesserung der Out-Of-Box-Experience in Saros mittels Heuristischer Evaluation und Usability-Tests. Master's thesis, Freie Universität Berlin, Institut für Informatik, 2011

[Bitterling2011] Patrick Bitterling, Design und Implementierung der neuen Saros Benutzeroberfläche, Bachelor's thesis, Freie Universität Berlin, Institut für Informatik, 2011

[Narweleit2011] Alexander Narweleit**,** Analyse und Evaluation von Verbesserungsmöglichkeiten der User Experience in Saros, Bachelor's thesis, Freie Universität Berlin, Institut für Informatik, 2011

[Nielsen1990] J. Nielsen,und R. Molich, Heuristic evaluation of user interfaces, ACM, 1990

[Nielsen1993] J. Nielsen, Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY. 1993

[Nielsen1993p] J. Nielsen, Iterative User Interface Design, Originally published in IEEE Computer Vol. 26, No. 11 (November 1993), pp. 32-41

[Nielsen1994] J. Nielsen, R. L. Mack, Usability Inspection Methods. New York, Wiley John + Sons, 1994

[McGeeRichDumas2004] Mick McGee et Al., UNDERSTANDING THE USABILITY CONSTRUCT: USER-PERCEIVED USBILITY in ROCEEDINGS of the HUMAN FACTORS AND ERGONOMICS SOCIETY 48th ANNUAL MEETING, Ph.D. Oracle Corporation Redwood Shores, CA, USA, Aaron Rich Oracle Corporation Redwood Shores, CA, USA, Joe Dumas, Ph.D. Oracle Corporation, Burlington, MA, USA, 2004

[ISO99] Norm DIN EN ISO 9241-1, Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten Teil 11: Anforderungen an die Gebrauchstauglichkeit – Leitsätze, 1999

[Moser2000] Karin S. Moser**,** Mentale Modelle und ihre Bedeutung, Kognitionspsychologische Grundlagen des (Miss-)Verstehens, in U. Ganz-Blättler, Michel, P. (Ed.), Sinnbildlich schief: Missgriffe bei Symbolgenese und Symbolgebrauch . Bern: Peter Lang. 2000

[Smith1986] Sidney L. Smith, Jane N. Mosier, "GUIDELINES FOR DESIGNING USER INTERFACE SOFTWARE", The MITRE Corporation- Bedford, Massachusetts, USA- Prepared for Deputy Commander for Development Plans and Support Systems, Electronic Systems Division, AFSC, United States Air Force, Hanscom Air Force Base, Massachusetts, 1986

[Gutwin1996] C. Gutwin, S. Greenberg, Workspace Awareness in Distributed Real-Time Groupware. PhD thesis, University of Calgary, Department of Computer Science, 1996.

[Gutwin2004] C. Gutwin, S. Greenberg. The Importance of Awareness for Team Cognition in Distributed Collaboration, In E. Salas, S. Fiore (Eds) Team Cognition: Understanding the Factors that Drive Process and Performance, 2004.

[Herczeg2005] M. Herczeg, „Software Ergonomie – Grundlagen der Mensch-Computer-Interaktion", 2. Auflage, Oldenburg Verlag, München, Wien, 2005

[Sarodnick2006] F. Sarodnick, H. Brau: Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung. Huber, Bern, 2006

[Dykstra1993] D. J. Dykstra, „A comparison of Heuristic Evaluation and Usability Testing: The Efficacy of a Domain-Specific Heuristic Checklist", Department of Industrial Engineering, Texas A&M University, Ph.D., 1993

[Holz1993] B. Holz auf der Heide, Welche software-ergonomischen Evaluationsverfahren können was leisten? in K.-H. Rödiger, Von der Benutzungsoberfläche zur Arbeitsgestaltung (Fachtagung des German Chapter of the ACM), 1993

[Dix2003] A. Dix, J.E. Finlay, et al."HumanComputer Interaction (3rd edition)", Prentice Hall, 2003

[Muraitis2001] A. Muraitis, Die Maschine liegt in der Natur des Menschen, PopScriptum 7 - Musik und Maschine, Forschungszentrum Populäre Musik der Humboldt-Universität zu Berlin, 2001

http://www2.hu-berlin.de/fpm/popscrip/themen/pst07/pst07_muraitis.htm, Abrufdatum 10.02.2012

[Rudolph2011] Carmen Rudolph, Evaluierung von Usability durch standardisierte qualitative Leitfadeninterviews, Master'S Thesis, Fakultät für Information, Technische Universität München, 2011

[Faulkner2003] L. Faulkner, Beyond the Five-User Assumption: Benefits of Increased Sample Sizes in Usabiltiy Testing, Behavior Research Methods, Instruments & Computers 35 (3), 2003

[Tidwell2011] J. Tidwell, Designing Interfaces, 2. Auflage, O'Reilly Media Inc., 2011

[Wharton1994] C. Wharton, J. Rieman, C. Lewis and P. Polson, The Cognitive Walkthrough Method: A Practitioner's Guide, in J. Nielsen and R. Mack (eds.), Usability Inspection Methods, John Wiley & Sons, Inc., New York, 1994, Ch. 5.

[CUP03] Hvannberg et Al., Classification of usability problems (CUP) scheme in Proceedings of INTERACT 2003

[SaadawiEtAl2005] Gilan M. Saadawi et Al. A Method for Automated Detection of Usability Problems from Client User Interface Events in AMIA Annu Symp Proc. 2005; 2005: 654–658.

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1560804/, Abrufdatum 10.01.2012

[Als2005] Benedikte S. Als, Janne J. Jensen, Mikael B. Skov, Comparison of Think-aloud and Constructive Interaction in Usability Testing with Children, Proceedings of the 2005 conference on Interaction design and children, Boulder, Colorado, 2005

[Haak2004] MJ Van den Haak, PJ Schellens, Employing think-aloud protocols and constructive interaction to test the usability of online library catalogues: a methodological comparison, in Interacting with Computers, Vol. 16, No. 6., 2004

[Wiedenbeck1999] Susan Wiedenbeck, "The use of icons and labels in an end user application program: an empirical study of learning and retention", in Behaviour & Information Technology, 18(2), p68-82, 1999

[Jetter2006] "Die MCI im Wandel: User Experience als die zentrale Herausforderung?" von Hans-Christian Jetter, Universität Konstanz, Arbeitsgruppe Mensch-Computer Interaktion, in A. M. Heinecke, H. Paul (Hrsg.): "Mensch & Computer 2006: Mensch und Computer im StrukturWandel". München, Oldenbourg Verlag, 2006

[Harrison2007] Harrison, C., Amento, B., Kuznetsov, S., and Bell, R.. Rethinking the progress bar. In Proceedings of the 20th Annual ACM Symposium on User interface Software and Technology (Newport, Rhode Island, USA, October 07 - 10, 2007). UIST '07. ACM, New York, NY. 115-118

[Frederickson1993] Fredrickson, B. L., and Kahneman, D. (1993). Duration Neglect in Retrospective Evaluations of Affective Episodes. Journal of Personality and Social Psychology, Vol. 65, pp. 45-55. Via [Harrison2007]

[Myers1985] B. A. Myers, The importance of percent-done progress indicators for computer-human interfaces. In Proceedings of the 1985 SIGCHI Conference on Human Factors in Computing Systems (San Francisco, California). CHI '85. ACM Press, New York, NY, pp. 11-17., 1985

[MicrosoftGuideline] http://msdn.microsoft.com/en-us/library/windows/desktop/aa511502.aspx, Abrufdatum 10.01.2012

[EclipseGuideline] http://wiki.eclipse.org/User_Interface_Guidelines, Abrufdatum 10.12.2011

[WikiUsability] http://de.wikipedia.org/wiki/Usability, Abrufdatum 20.02.2012

[WikiIso9241] http://de.wikipedia.org/wiki/EN_ISO_9241, Abrufdatum 10.01.2012

[WikiGroupware] http://de.wikipedia.org/wiki/Groupware, Abrufdatum 26.03.2012

[WikiPriming] http://de.wikipedia.org/wiki/Priming_(Psychologie), Abrufdatum 12.01.2012

[WikiWidescreen] http://en.wikipedia.org/wiki/Widescreen, Abrufdatum 12.01.2012

[Nielsen2000] J. Nielsen, Why you only need to test with 5 users, 19.03.2000, Alertbox http://www.useit.com/alertbox/20000319.html, Abrufdatum 15.12.2011

[SarosWeb1] http://www.saros-project.org/Research, Abrufdatum 20.02.2012

[SarosWeb2] http://www.inf.fu-berlin.de/w/SE/ThesesDPP, Abrufdatum 20.02.2012

[NielsenWeb1] http://www.useit.com/jakob/, Abrufdatum 10.02.2012

[Nielsen2005] J.Nielsen, Severity Ratings for Usability Problems, http://www.useit.com/papers/heuristic/severityrating.html, Abrufdatum 10.01.2012

[Nielsen2010] Jakob Nielsen's Alertbox, October 18, 2010, http://www.useit.com/alertbox/mental-models.html, Abrufdatum 10.02.2012

[UmfragenWeb] http://blog.net-corps.de/diplom/surveys.txt, Bereinigte Kopie der Rohdaten im CSV-Format, ohne IP-Adressen und ohne Emailadressen

Windows Mauszeiger Icons von http://www.phodana.de/resources/freebie-mauszeiger-als-vektorgrafik/

Anmerkung: zu [Gutwin2004] existieren unterschiedliche Jahresangaben. Siehe http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.3674 (2001 vs. 2004)

Abbildungsverzeichnis

Abbildung 1: Eclipse Entwicklungsumgebung mit Package Explorer (1), dem Editor-Bereich (2) sowie dem Saros-View (3) mit der Buddyliste (links). 6
Abbildung 2: Übersicht der Saros-Architektur (Quelle: www.saros-project.org) 13
Abbildung 3: Screenshots der beiden Seiten des Einladungs-Wizards (Ressourcen und Teilnehmer) 14
Abbildung 4: Eclipse-Ansicht in einer laufenden Saros-Sitzung mit farbig markierten Änderungen durch einen der Sitzungsteilnehmer 14
Abbildung 5: Verhältnis zwischen Anzahl der Nutzer und Prozentsatz der gefundenen Probleme, Jakob Nielsen‘s Alertbox, 19.03.2000 (Quelle: http://www.useit.com/alertbox/20000319.html) 29
Abbildung 6: Usability Probleme nach Heuristik-Verletzungen 50
Abbildung 7: Usability Probleme nach Funktionsbereichen 50
Abbildung 8: Der Getting Started Wizard 52
Abbildung 9: Die Windows-Einstellung „Mittel – 125%“ 52
Abbildung 10: Links die abgeschnittene Darstellung, Rechts die korrekte Darstellung des Wizards 53
Abbildung 11: Initiale Darstellung des Dialogs zum Hinzufügen eines Buddies 53
Abbildung 12: Hinweise auf Eigenheiten des öffentlichen Saros-Servers 54
Abbildung 13: Technischer Dialog mit Rechtschreibfehlern und Fachbegriffen 55
Abbildung 14: Technischer Dialog für das Bestätigen des Hinzufügens eines Buddies 55
Abbildung 15: Ablauf eines Einladungsprozesses 56
Abbildung 16: Menüs zum Starten einer Sitzung, Quelle www.saros-project.org 57
Abbildung 17: Die Wizard-Seite zum Auswählen von Ressourcen für die Sitzung 57
Abbildung 18: Die Auswahl der Sitzungsteilnehmer 58
Abbildung 19: Die Ansicht für den Host nach Beenden des Wizards 59
Abbildung 20: Einladung zu einer Sitzung. (1) Text der Einladung, (2) Hinweistext in der Wizard-Kopfzeile 60
Abbildung 21: Editoren die aus mehreren Teilen bestehen, werden nicht synchronisiert 61
Abbildung 22: Benachrichtigung wenn eine Datei geöffnet ist im Follow-Modus 61

Abbildung 23: Der Package Explorer mit Icon-Markierungen 62
Abbildung 24: Mangelnde Ästhetik und Handhabung der Ballon-Benachrichtigungen 63
Abbildung 25: Anteil der Frauen in den Studiengängen der Informatik, 69
Abbildung 26: Darstellung eines Fortschrittsbalkens für einen Prozess mit unbekannter Dauer 75
Abbildung 27: Linearer Verlauf des Fortschrittsbalkens für die Checksummen-Berechnung 75
Abbildung 28: Wartezustand beim Host 76
Abbildung 29: Wartezustand beim Empfänger 76
Abbildung 30: Detaillierte Fortschrittsanzeige beim Host 77
Abbildung 31: Detaillierte Fortschrittsanzeige beim Empfänger 77
Abbildung 32: Ein Hinweis darauf, dass die Ermittlung der SVN Revisionen Zeit kostet 78
Abbildung 33: Gespiegelte Fortschrittsanzeige für Host und eingeladenen Nutzer bei einem SVN Checkout über Saros 78
Abbildung 34: Der alte Wizard für die Projektselektion 80
Abbildung 35: Der neue Wizard zum Auswählen der Ressourcen für das Projekt 81
Abbildung 36: Layout-Entwurf für Projekt-Selektion im Einladungs-Wizard 83
Abbildung 37: Alte Auswahl der Buddies mit allen Nutzern 83
Abbildung 38: Die überarbeitete Auswahl der Buddies 84
Abbildung 39: Zusammenfassung der Einladung als letzte Seite im Wizard 85
Abbildung 40: Die überarbeitete Auswahl der Sitzungsteilnehmer 87
Abbildung 41: Der Wizard für eingehende Projekte nach Annahme einer Einladung 88
Abbildung 42: der Überarbeitete Wizard für eingehende Projekte 91
Abbildung 43: Wizard für eingehende Projekte (eines mit SVN Bindung) 91
Abbildung 44: Der alte Saros-View 93
Abbildung 45: Der überarbeitete Saros-View mit mehr Awareness-Informationen 94
Abbildung 46: Nahaufnahme der Baumdarstellung für die Sitzung 94
Abbildung 47: Indikation des aktiven Follow-Modus 95
Abbildung 48: Indikation des Follow-Modus anderer Sitzungsteilnehmer 95
Abbildung 49: Alter Saros-View mit Alias und XMPP ID in Klammern 96
Abbildung 50: Indikation der HOST-Rolle für den Nutzer „netcorpse“ 97
Abbildung 51: Markierung eines partiell geteilten Projekts 97
Abbildung 52: Ballon-Benachrichtigung für das Beitreten eines Nutzers zur Sitzung 98
Abbildung 53: Der Eclipse Welcome Screen 99
Abbildung 54: Das neue Getting Started in Eclipse 100
Abbildung 55: Das Dashboard im Saros-View 101
Abbildung 56: Vorgegebene Kontextmenü-Reihenfolge aus den Eclipse-Guidelines [EclipseGuideline] 103
Abbildung 57: Kontextmenüs vor der Überarbeitung 104
Abbildung 58: Kontextmenüs nach der Überarbeitung 104
Abbildung 59: Listener für das Changelog 105
Abbildung 60: Der Saros-View: Hier wäre Platz für die Sitzungshistorie 106
Abbildung 61: Verbesserung der Awareness durch Übermittlung von Dialogtiteln 107
Abbildung 62: Screenshot aus der Durchführung des Akzeptanztests 108
Abbildung 63: Verteilung der offenen Probleme nach Heuristiken 112
Abbildung 64: Verteilung der offenen Probleme nach Funktionsgruppen 112
Abbildung 65: Übersicht der Usability Verfahren mit Vorteilen und nötiger Probanden-Anzahl [Nielsen1993, S.224, Table 10 Summary of the usability methods] 117
Abbildung 66: Fortschrittsanzeige eines User-Jobs 120
Abbildung 67: Zwei zusammengefasste Einladungsprozesse 122
Abbildung 68: Eclipse Progress View 123
Abbildung 69: Fortschrittsbalken in einem Wizard (Bildquelle: www.eclipse-tips.com) 123