Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Software Engineering

Evaluation der Mechanismen zum Darstellen der Workspace Awareness in Saros

Arsenij E. Solovjev
Matrikelnummer: 4223897
xeper@inf.fu-berlin.de

Betreuer: Julia Schenk
Eingereicht bei: Prof. Dr. Lutz Prechelt
Zweitgutachter: Prof. Dr. Elfriede Fehr

Berlin, 05.09.2011

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

05.09.2011

Arsenij E. Solovjev

Zusammenfassung

Eine der kritischsten Aspekte von Software zur verteilten Paarprogrammierung oder verteilten Softwareentwicklung allgemein betrifft die Möglichkeiten eines Nutzers über den aktuellen Ort innerhalb des Codes und die aktuellen Aktivitäten der anderen Nutzer auf dem Laufenden zu sein. Sieht z.B. der andere Nutzer gerade das, was ich ihm erklären möchte? Wer schon mal Fernwartung übers Telefon gemacht hat, weiß, wie wichtig es ist, dass man ein gutes Verständnis für das hat, was beim Anderen vorgeht. Dieses Verständnis bzgl. Anderer und deren Aktivitäten in einer gemeinsamen Umgebung wird als Awareness bezeichnet. Saros ist bereits mit vielen Mechanismen zum Darstellen von Awarenessinformationen ausgestattet. Es ist aber festgestellt worden, dass das Maß an Präsenz noch deutlich hinter der klassischen (nicht verteilten) Paarprogrammierung liegt. Ziel dieser Arbeit ist es, die Awarenessmechanismen in Saros zu analysieren und zu evaluieren. Es wurden Aufzeichungen von der Arbeit mit Saros analysiert, und in diesen Ereignisse identifiziert, in denen Awarenessinformationen gut, schlecht oder gar nicht übertragen worden sind. Die Analyse befolgte ansatzweise die Grounded Theory Methodology. Das Ergebniss ist ein Überblick über die Problemen und Stärken von Saros in der Übertragung von Awarenessinformationen.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangssituation

Eine der kritischsten Aspekte von Software zur verteilten Paarprogrammierung oder verteilten Softwareentwicklung allgemein betrifft die Möglichkeiten eines Nutzers über den aktuellen Ort innerhalb des Codes und die aktuellen Aktivitäten der anderen Nutzer auf dem Laufenden zu sein. Sieht z.B. der andere Nutzer gerade das, was ich ihm erklären möchte? Wer schon mal Fernwartung übers Telefon gemacht hat, weiß, wie wichtig es ist, dass man ein gutes Verständnis für das hat, was beim Anderen vorgeht. Dieses Verständnis bzgl. Anderer und deren Aktivitäten in einer gemeinsamen Umgebung wird als Awareness bezeichnet.

1.2 Problem

Die Groupware für verteilte Softwareentwicklung die hier untersucht wird ist Saros. Saros ist bereits mit vielen solchen Mechanismen ausgestattet. Es ist jedoch festgestellt worden, dass diese nicht die Fülle an Informationen abbilden die bei der nichtverteilten Softwareentwicklung als selbstverständlich angenommen wird.

1.3 Ziel

Ziel dieser Arbeit ist es, ein Überblick über die Awarenessmechanismen in Saros zu verschaffen. Ferner werden diese unter dem Gesichtspunkt, welche Awarenessinformationen übertragen werden, und noch interessanter welche nicht übertragen werden, ausgewertet.

1.4 Methode

Um dieses Ziel zu erlangen sind Aufzeichnungen von Saros-Sitzung aus einem industriellem Betrieb gemacht worden. Diese wurden mittels Verwendung von Ansätzen und Techniken aus der Grounded Theory Methodology analysiert.

1.5 Aufbau dieser Arbeit

Im Kapitel 2 werden die Begriffe der Awarenes und der Workspace Awareness definiert und erörtert. Ferner wird Saros als eine Groupware die Distributed Pair Programming unterstützt vorgestellt, und es werden genauer die bestehenden Workspace Awareness Mechanismen(WAM) eingegangen. Im Kapitel 3 wird die Methodik vorgestellt, die Art wie die Aufzeichnungen gemacht worden sind, so wie die eigentliche Auswertung der WAMs in Saros. Kapitel 4 befasst sich mit der Erfahrung der Datenerehebung im industriellen Betrieb, und macht ein Paar Schlüsse daraus. Kapitel 5 bezieht sich

auf die Erfahrung der Mitarbeit im Saros-Entwicklungsteam. Anschliessend ist im Kapitel 6 der Fazit und der Ausblick zu finden.

2 Grundlagen

In diesem Kapitel werden werden die Grundlagen dieser Arbeit dargestellt. Als erstes gehe ich auf dem Begriff der Workspace Awareness (WA) ein, und erläutere das Konzept von Distributed Pair Programming (DPP) erläutert. Danache Tool welches DPP unterstützt und Gegenstand dieser Arbeit ist, Saros. Im letzten Teil werden im Detail die Mechanismen vorgestellt die die WA in Saros darstellen vorgestellt.

2.1 Awareness

Nun setzen wir uns mit dem zentralen Begriff dieser Arbeit, der Awareness, auseinander und befassen uns in diesem Zuge und damit was Shared Workspace und was Workspace Awareness sind. Im Rahmen der Dissertation "Workspace Awareness in Distributed Real-Time Groupware" setzen sich Gutwin und Greenberg mit dem Begriff der Awareness auseinander. Sie stellen fest, dass diese kann in Hinsicht auf die zwei Begriffe, Wissen und Bewusstsein betrachtet werden kann. In der Dissertation wird sie unter dem angemesseneren Gesichtspunkt des Wissens betrachtet. So stellen Gutwin und Greenberg folgende Eigenschafen für diese fest [1]:

1. Awareness ist Wissen über den aktuellen Zustand einer Umgebung, ein Rahmen der von Zeit und Raum beschränkt ist

2. Umgebungen verändern sich über Zeit, so ist Awareness Wissen welches gewartet und auf dem aktuellen Zustand gehalten werden muss.

3. Menschen interagieren mit der Umgebung und die Wartung der Awareness geschieht durch diese Interaktion.

4. Awareness ist fast immer ein Teil einer anderen Aktivität. Das heißt, Awareness ist selten Ziel einer Aktivität, das Ziel ist die gegebene Aufgabe zu erfüllen, bei dessen Ausführung Awareness eine Rolle spielt.

2.1.1 Shared Workspaces

Eine Shared Workspace ist kurz gefasst ein gemeinsam genutzter physischer Arbeitsbereich, in dem Leute eine gemeinsame Aktivität betreiben. Die Charakteristiken welche eine Shared Workspace ausmachen, und die einen Einfluss auf die Awareness ausüben, sind offensichtliche Verfügbarkeit, deren räumliche Organisation und begrenzte Interpretation.

[1] wörtliche Übersetzung vom Englischen aus [GG96]

2.1.2 Situation Awareness

Situation Awareness (SA) ist ein Begriff aus der Arbeitswissenschaft der dem Begriff der in dieser Arbeit welcher in dieser Arbeit verwendet wird am nähesten kommt. Mit SA wird zum Beispiel Awareness eines Piloten während eines Flugs, in der Anesthäsiologie oder in der Flugverkehrskontrolle untersucht, wo die Awareness eine kritische Rolle spielt.
Die Situation Awareness hat Aspekte eines Prozesses und eines Produkts. Als Prozess umfasst sie das der Wahrnehmen, das Verstehen und die anschliessende Untersuchung der gegebenen Umgebung. Durch das Wahrnehmen und das Verstehen entsteht Wissen. Ferner steuert dieses Wissen die Richtung der Untersuchung in einem gewissen Maß.
Die Beziehung zwischen Wissen, Umgebung und Untersuchung, kann mit folgendem Diagramm dargestellt werden :

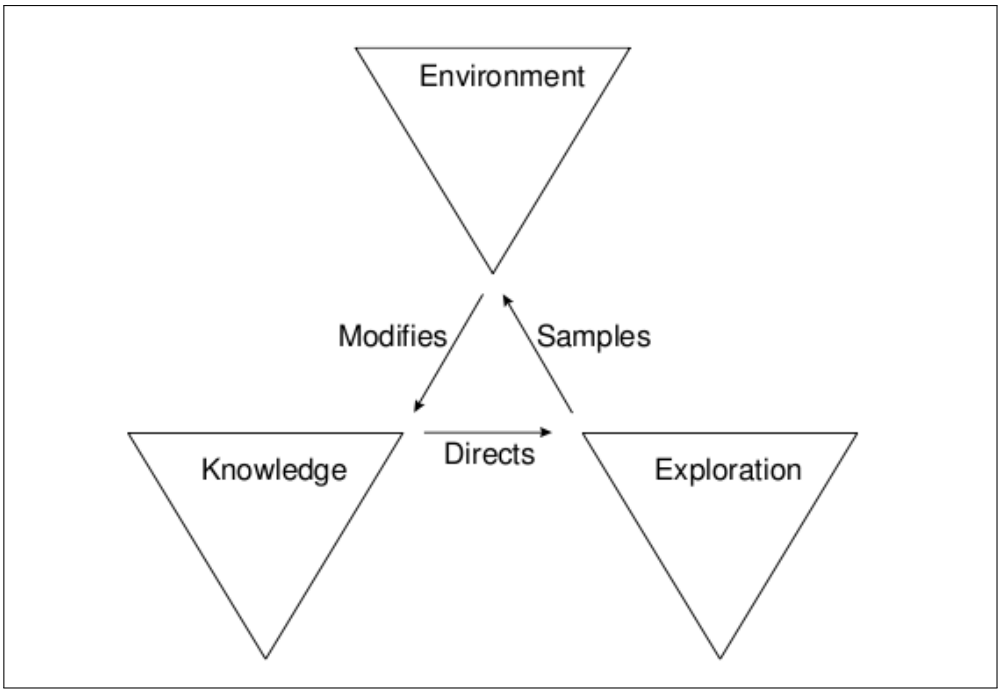


Abbildung 1: Der *perception-action cycle* (Neisser 1976)

2.1.3 Workspace Awareness

SA hat verschiedene Ausprägungen, z. B. die Spatial Awareness, bei der der Pilot weiß wo er sich im Flugraum befindet. Workspace Awareness ist in diesem Fall eine Spezialisierung der Awareness von Anderen bei Zusamme-

narbeit welche selbst eine Art der SA ist.

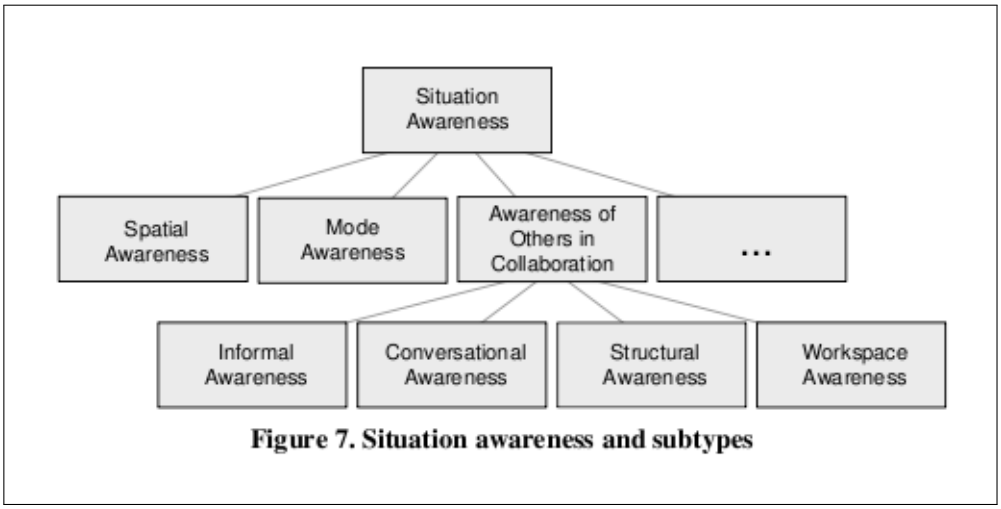


Abbildung 2: Die Beziehung zwischen SA und WA

Über Workspace Awareness(WA) sagt Gutwin "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"[GG96]. Dazu sei gesagt, dass Workspace Awareness sich auf mittelgroße Shared Wokrspaces beschränkt und sich sich auf die Kollaboration mit Anderen beschränkt.

2.2 Distributed Party Programming

Im vorherigem Abschnitt wurde der Begriff der Workspace Awareness erörtert. Bevor vorgestellt wird wie diese in Saros dargestellt wird, muss erst der Begriff des Distributed Party Programmings erörtert werden. Dazu werden hier aneinander anschliessend die Begriffe Pair Programming(PP), Distributed PP, und Distributed Party Programming(DPP) erörtert.

Pair Programming ist eine agile Entwicklungstechnik bei der zwei Entwickler nebeneinander sitzen und an einer Aufgabe arbeiten, mit dem Ziel dadurch eine bessere Code-Qualität zu gewinnen. Dabei findet eine Rollenverteilung statt, der eine Entwickler nimmt die Rolle des Drivers, er bedient die Tastatur und die Maus, und erfüllt physisch die gegebene Aufgabe. Der Observer hingegen beobachtet den Driver und macht eine Review während der Driver an der Aufgabe arbeitet. So kann sich der Driver an die Ausführung der Aufgabe konzentrieren, während der Observer den Überblick behält, die Richtung der Arbeit steuert und neue Ideen beisteuert.

Distributed Pair Programming geschieht wenn zwei Entwickler mit Hilfe von Groupware PP betreiben, räumlich aber getrennt sind.
Distributed Party Programming ist eine Verallgemeinerung von Distributed PP. Im Unterschied zu Distributed PP ist die Anzahl der Teilnehmer nicht nur auf zwei beschränkt , es gibt keine strikte Rollenverteilung, d.h. man kann Driver oder Observer werden, muss man aber nicht, man kann auch unabhängig von den anderen Teilnehmern arbeiten.

2.3 Saros

Das im vorherigem Abschnitt beschriebenen Distributed Party Programming wird von Saros unterstüzt. Saros ist ein Open-Source Eclipse-Plug-in, welchen Riad Djemili im Rahmen seiner Diplomarbeit an der Freien Universität entwickelt hat[Dje06]. Diese wurde im Rahmen von anderen wissenschaftlichen Arbeiten und von Mitglieder der AG Software Engineering, der Freien Universität in Berlin, weiter entwickelt. Saros realisiert folgende Aspekte, um DPP möglich zu machen:

2.3.1 Synchronization

Saros ermöglicht den Entwicklern in einer Session mit einander zu treten und auf einen synchronisierten und sich synchronisierenden Datenstand zu arbeiten. Dies geschieht indem Entwickler A einem Entwickler B eine Einladung auf einen Projekt zu arbeiten verschickt. Nach dem die Einladung akkzeptiert wurde, werden die Workspaces von A und B auf das gegebene Projekt (oder auf einen Teil eines Projektes) synchronisiert. Dies kann durch Kopieren des Projektes in das Workspace von B oder durch synchronisieren einer bestehnden Kopie des Projektes bei B geschehen. Dies hat als Auswirkung, dass wenn A eine Datei verändert, B automatisch dieselbe Änderung im eigenen Workspace bekommt.

2.3.2 Driver-Observer

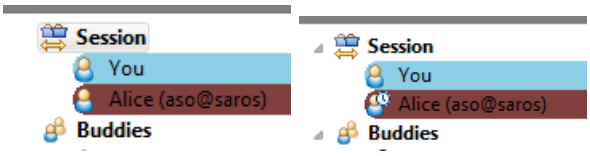
Saros implementiert ausserdem das Rollen-Konzept aus der klassischen Paar-Programmierung (Driver-Observer). Ein Buddy[1] kann in einen "Follow-Mode" gehen in welchem er automatisch die Sicht des anderen Buddies mitverfolgt, d.h er sieht immer den gleichen Abschnitt der gleichen Datei. Bewegt sich der andere Buddy in der Datei, oder springt er in eine andere Datei, springt die Ansicht des folgenden Buddys (des Observers) automatisch mit. Im Unterschied zu klassischen Paar Programmierung, können innerhalb einer Session mehrere Observer einem Driver folgen.

[1] in Saros hält man eine Liste von Leuten mit denen man kollaborieren kann, diese werden Buddies genannt

2.3.3 Bestehende Workspace Awareness Mechanismen in Saros

Zum Zweck des Aufrechterhaltens der WA sind in Saros, Version 11.7.29.r3479, folgende Workspace Awareness Mechanismen(WAM) realisiert:

- In der Rosterview wird die Buddyliste angezeigt, samt der Information wer von den Buddies online und zu einer Saros Session eingeladen werden kann, und wer von denen Offline ist. Es werden zusätzliche Information, wie zum Beispiel ob mit dem betroffenen Buddy eine IBB oder Socks5 Verbindung besteht. Zusätzlich sind die Teilnehmer einer Session dargestellt, samt deren zugeordneten Farbe. Bei einem selber wird angegeben ob man in Follow-Mode ist oder nicht. Ferner, damit klar wird ob der andere Teilnehmer einer Sitzung gerade seine Aufmerksamkeit auf die Shared Wokrspace richtet, wird wenn ein Buddy das Eclipse Fenster nicht im Vordergrund hat, seiner Icon in der Rosterview ein Uhr hinzugefügt.



- Viewports markieren mit der Farbe des jeweiligen Buddys, welchen Bereich der gerade geöffneten Datei dieser sieht. Es werden einmal der Bereich in der aktuellen Sicht markiert, so wie der Bereich in der Datei überhaupt.

Abbildung 3: Viewports in Saros

- Wenn ein Buddy Code markiert, so wird der markierte Bereich in dessen Farbe beim Gegenüber markiert genau so lange wie dieser die

Markierung aufrechterhält. Somit kann der Entwickler wichtige Stellen beim Erklären hervorheben ohne dabei explizit auf diese hinzudeuten, weder durch Benennung der Stelle noch durch expliziter Zeilenangabe.

- Die Teile der Datei die vom Buddy editiert worden sind werden farblich markiert so lange die Session läuft, so kann während der Session der Author des Codes ermittelt werden, und falls nötig drauf angesprochen werden.

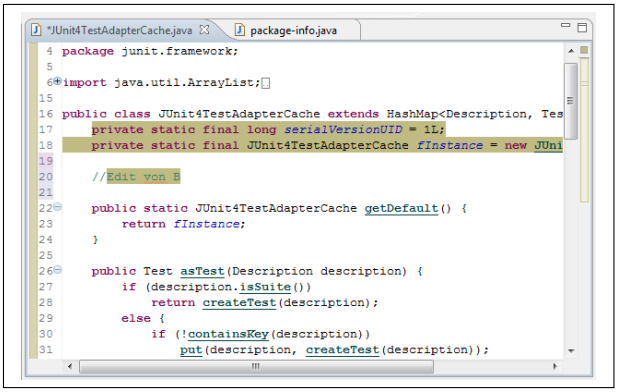


Abbildung 4: Markierungen in Saros

- Die Position des Zeigers wird ebenso farblich gekennzeichnet

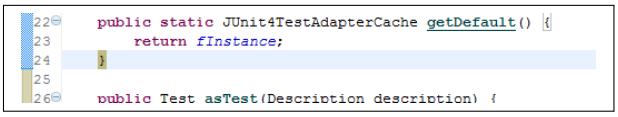


Abbildung 5: Der markierte Zeiger in Saros

- Der Follow-Mode ist ein weiteres WAM, der die aktuelle Position des Driver automatisch mitverfolgt. Scrollt dieser hoch und runter, so bewegt sich die Sicht ebenso beim Observer. Wenn der Driver eine andere Datei öffnet wird diese auch beim Observer automatisch geöffnet.

- Awareness während des Einladungs und Synchronisierungsprozesses. Dies umfasst alle Dialoge, Progress Bars die während dem Aufbau der Session eine Rolle spielen. Zum Beispiel liefert der Einladungsdialog beim Einladenden, die Information dass der Buddy an einem Projekt arbeiten möchte.

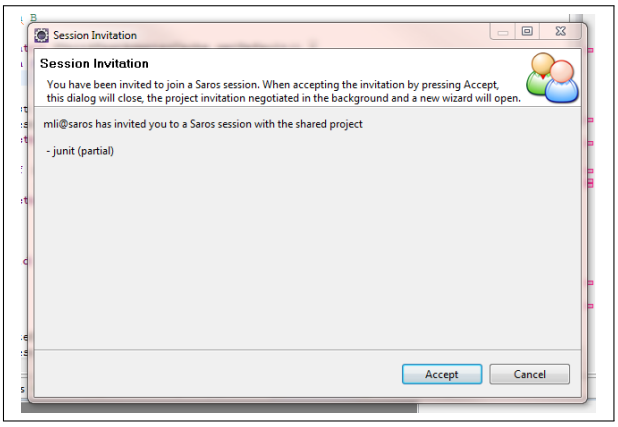


Abbildung 6: Der markierte Zeiger in Saros

- Saros bietet die Möglichkeit Screensharing zu machen. Ein Entwickler kann entweder den gesamten Bildschirminhalt des Buddies einsehen, oder einen bestimmten Bereich um den Zeiger mitverfolgen. Leider ist dieses Feature wegen unausgereifetem Stadiums aus der Studie rausgefallen. Hier sei es aber Vollständigkeitshalber erwähnt.
- Es werden je nach Ereigniss auch verschiedene Benachrichitgungen ausgegeben. Diese sind:
  - ”Buddy joined the session“
  - ”Buddy closed the session”
  - “Inconsistency detected” taucht auf wenn zwei synchronisierte Dateien nicht auf dem selben Stand sind.

2.3.4 Szenarien

In [SS10] identifizieren die Autoren folgende Szenarien bei denen Saros zur Anwendung kommen kann. Dabei wird angenommen, dass die beteiligten Benutzer über eine VoIP Verbindung verfügen, um sich während der Aufgabe über diese untereinander auszutauschen. Dies Szenarien sind:

CodeWalkthrough Bei der Einführung neuer Entwickler in einem Team, kann es von Nöten sein denen eine Einführung in dem Quellcode zu geben. Dabei agiert ein in der Regel erfahrenerer Entwickler als Driver, und die Entwickler könne ihm im Follow-Mode folgen, als dieser die interessanten Stellen im Quellcode erklärt.

CodeReview In diesem Szenario, wollen Entwickler eine Durchsicht an einer Quellcodeeinheit, um nach Defekten und Schwächen darin zu

suchen. Im Unterschied zu *CodeWalkthrough* wollen alle Beteiligten inhaltlich was beibringen, so gibt es Perioden wenn die Beteligten zusammen ein Stück Quellcode untersuchen und Perioden wenn sie unabhängig voneinander analysieren. Saros eignet sich idealerweise dafür, da man frei aus dem Follow-Mode rein- und raustreten kann.

Distributed PP Wollen zwei Entwickler eine PP Sitzung machen ohne dabei am selben Ort zu sein, biete Saros die Möglichkeit dies verteilt zu machen. So lädt der eine Teilnehmer den Anderen zu einer Saros Sitzung ein, dabei nimmt der eine die Rolle des Driver und der tritt in dem Follow-Mode ein, und nimmt dadurch die Rolle des Observer. Der Rollentausch kann durch rein- und raustreten aus dem Follow-Mode stattfinden.

DPP Mehrere Entwickler arbeiten gleichzeitig unabhängig voneinander. Durch die Rosterview sind sie von der Anwesenheit untereinander informiert. Möchte einer der Entwickler sich über einen Problem mit einem anderen austauschen, so kann er eine Saros-Session mit diesem starten, und sich über die betroffene Stelle im Quellcode mit ihm austauschen.

3 Qualitative Datenanalyse

Im vorherigem Kapitel wurden Saros, die Begriffe der Awareness und der Workspace Awareness, so wie die in Saros bereits umgesetzten Mechanismen die dazu dienen die letztere abzubilden, vorgestellt. Nun kennen wir den Gegenstand unserer Evaluation und wir können uns nun mit der verwendeten Methodik vertraut machen.
In Unterkapitel 1 stelle ich die Methodik der Evaluation vor und im Zweiten den Ursprung und die Art der Daten mit denen ich gearbeitet habe. Das dritte Unterkapitel erläutert die Interviews und das Vierte präsentiert die Ergebnisse dieser Arbeit.

3.1 Methodik

3.1.1 Grounded Theory

Grounded Theory Methodology (GTM) ist eine Vorgehensweise, die darauf zielt "durch qualitative Datenanalyse induktiv Theorien zu erstellen" [uJC96]. Mit "induktiv" ist hier gemeint, dass von den Daten ausgehend eine Theorie erzeugt wird. Sie wurde von Glaser und Strauss entwickelt, und die erste Veröffentlichung darüber ist aus dem Jahre 1967 und heißt "The Discovery Of Grounded Theory". Obwohl die GTM aus der Soziologie kommt, hat sich die gut für empirische Forschung im Software Engineering bewährt [Car07]. Das Ziel von der Grounded Theory ist eine Theorie zu erstellen die "grounded" ist, im Deutschen wird dies häufig als gegenstandsverankerte Theoriebildung bezeichnet. Gegenstandsverankert soll hier soviel heißen, dass die daraus entstehende Theore ihren Ursprung in den Daten hat. Die gewonnene Theorie soll den Kriterien der Übereinstimmung, Verständlichkeit, Allgemeingültigkeit und Kontrolle genügen. Was die GTM darstellt ist eine wissenschaftliche Methodologie um diese Ziele zu erlangen, die aus einer Reieh von Richtlinien und Techniken besteht. Ein wichtiges Merkmal ist, dass diese nicht in einer strikt vorgegebenen Reihenfolge stattfinden, sondern immer abwechselend, so dass diese einander beeinflussen können und Synergien entstehen welche der Theorie Tiefe und Präzision verleihen. Die wichtigsten dieser Techniken sind das Kodieren, das theoretische Sampling, das ständige Vergleichen sowie die Memos.

Kodieren ist das Konzeptualisieren der Daten, Phänomenen aus den Daten werden sogenannte Konzepte zugewiesen, die uns dann verhelfen über die Daten auf einer abstrakteren und für die Theoriebildung leichter handhabbaren Ebene nach zu denken. Einzelnen Vorkomnissen eines Phänomens werden sogenannte Kodes zugewiesen Es wird zwischen offenem, axialem und selektivem Kodieren unterschieden. Ein wichtiges

Merkmal, ist das diese drei Arten des Kodierens nicht in Phasen stattfinden, sondern parallel und beeinflußen einander.

offenes Kodieren beschreibt die Daten durch konzeptuelle Kodes welche unmittelbar aus den Daten stammen.

axiales Kodieren setzt die Kodes in eine Beziehung miteinander.

selektives Kodieren hier werden bestimmte Kodes und dessen Beziehungen entnommen, zwecks aus diesen eine zusammenhängende Theorie zu bilden.

Theoretisches Sampling ist die Art der Datenerhebung der iterativ im Zusammenhang mit dem Kodieren stattfindet. Dabei steuern Erkenntnisse aus der Analyse der bereits vorhandenen Daten die Erhebung der nächsten Daten.

Kontinuierlicher Vergleich ist das wiederholte Inbeziehung setzen und Vergleichen der kodierten Konzepte, zwecks Kodes zu erzeugen die konsistent und präzise sind[SS07].

Memos sind freiförmige Notizen die man während der Analyse erstellt, in denen wichtige Beziehungen zwischen Konzepte und Kategorien so wie Ideen etc. notiert werden. Dies dient dazu Ideen zu generieren welche der Theorie im Endeffekt Originalität und Tiefe verleihen.

Von dem Forscher wird dabei eine Fertigkeit, die sogenannte theoretische Sensibilität gefordert. Theoretische Sensibilität beinhaltet die Fähigkeit eine abstrakte statt eine deskriptive Interpretation aus den untersuchten Daten zu machen. Es ist eine Art der Interpretation, welche für weitere Theoriebildung erlaubt.

Obwohl eine gut durchgeführte Grounded Theory wissenschaftlich begründet ist, ist sie keineswegs fertig, da die Konzepte und Kategorien nie engültig fertig sein können, da man immer die Möglichkeit hat Konzepte und Kategorien in Beziehungen miteinander zu setzen. Es liegt immer am Forscher zu entscheiden wie tief und gründlich er sie betreiben möchte.

3.2 Aufzeichnungen und Datenerhebung

Als rohe Daten für die Analyse wurden Aufzeichnungen aus der Benutzung von Saros in einer echten Arbeistumgebung gemacht. Dafür habe ich mit meinem Arbeitgeber vereinbart, für meine Bachelorarbeit von einigen Entwicklern, bei passenden Aufagaben, Daten erhoben werden, die ich mnun für meine Bachelorarbeit verwende. In einem Gespräch mit dem Vorstand und dem damaligen Entwicklungsleiter, haben wir uns darauf geeinigt, dass

die Firma mir für die sechs bis acht Sessions je zwei oder drei Stunden lang die Unterstützung des Entwicklungsteams bereitstellt.

3.2.1 Entwicklungsumgebung und Art der Aufgaben

Da es von Interesse ist in was für eine Umgebung Saros verwendet worden ist, werden hier die verwendeten Werkzeuge, so wie die Arten der Aufgaben beschrieben. Die mittelgroße Firma entwickelt ein web-basiertes Übersetzungs- und Übersetzungsmanagementstool, bei welchem Technologien wie Seam, JBoss, JSF verwendet werden. Als Versionsverwaltungstool wird SVN verwendet und zwar sowohl als Eclipse-Plugin als auch die Windows Explorer Integration dessen, TortoiseSVN. Als Build-Tool wird Ant verwendet. Zusätzlich bedienten sich die Entwickler dem Browser, um das Tool zu testen. Skype wird zur Kommunikation verwendet, manchmal während der Aufzeichnungen hatten die Entwickler die Screensharing-Funktionalität dessen benutzt.

Die an der Studie beteiligten Entwicklern fanden sich während den Aufzeichnung meistens im selben Gebäude, öfters sogar im selben Raum, manchmal haben die auch von Zuhause aus gearbeitet. Die beteilgiten waren Festangestellte, Studenten und Freelancer, mit unterschiedlichem Erfahrungsgrad als Entwickler allgemein, und in der Firma. PP wurde vor der Ankündigung dieser Arbeit nur selten praktiziert, und wenn, wurde nicht als solches erkannt, “Wir haben uns mit X einfach zusammengesetzt und an dem Feature Y an einem Rechner gearbeitet”. war die Aussage einer der Entwickler.

Beinahe gleichzeitig mit dem Anfang dieser Arbeit jedoch unabhängig davon wurde die Wichtigkeit von Wissenstransfer innerhalb des gesamten Entwicklerteams stark betont. Dabei wurde PP, als eins der Mittel für Wissenstransfer, erkannt. Bei den analysierten Aufzeichnungen hatten alle Beteiligten, mit einer Ausnahme, bereits an einer Saros Session teilgenommen.

Ich habe keinerlei Vorgaben an die Entwickler gegeben, die Aufgaben die aufzuzeichnen waren, waren von den Entwicklern meistens selbst gewählt. Wenn der Leser mehr über die Datenerhebung während dieser Arbeit erfahren möchte, sei er auf Kapitel 5 verwiesen.

3.2.2 Was wurde aufgezeichnet?

Eine Aufzeichnung hatte folgende Bestandteile:

- Bildschirmaufnahme die den Eclipse beinhaltet,

- Audiospur des Gespräches mit dem anderen Entwickler in der Session, so wie eine
- Aufnahme des Entwicklers mit einer Webcam.

Da unser Ziel war zu sehen, wo die Stellen sind in der Benutzung, wo man auf Probleme bezüglich der Workspace Awarenes Informationen(WAIs) stoßen, war es wichtig, dass man auch die Mimik der Entwickler analysieren kann, um Hinweise zu bekommen ob ein Mißverständnis vorgelegen haben könnte. Ferner ist Awareness ein schwe zu erfassendes Phänomen, welches sich auf viele verschiedene Arten zeigen kann. Daher war es angebracht, so viel wie möglich von dem Verhalten, der Gestik und der Mimik der Entwickler in die Analyse mit einzubeziehen.

3.2.3 Womit wurden die Aufzeichnungen gemacht?

Die Software für die Aufzeichnungen hatte also die Anforderungen Mikrofon, Bildschirm und Webcam aufnehmen können. Es wurden mehrere freeware/OpenSource Tools ausprobiert, darunter CamStudio, Jing, AviScreen und Hypercam. CamStudio ist ein Paar mal einfach gecrasht, AviScreen konnte mit langen Videos nicht umgehen und hat zu große Videos produziert. Jing hatte eine Beschränkung auf 5 Minuten was nicht akkzeptabel war. Der ursprünglich vorgeschlagenen Screencast-o-matic wurde verworfen auch aus dem Grund, dass es eine zeitliche Begrenzung von einer Stunde für die Videos gab.
Letzendlich habe ich mich auf die 30-tägige Testversion von Camtasia Studio 7 entschieden, weil diese keine zeitliche Einschränkung hat, Mikrofon, Webcam und Screencast aufnimmt und diese in ein handhabbares Format exportieren kann. Als Fallback hatte ich mich für eine getrennte Aufnahme des Bildschirms mit VLC, und Aufnahme von Mikrofon und Webcam mit dem Windows Audiorecorder und dem
Durch organistorische Begebenheiten in dem Betrieb der Firma, wurden in dem geplantem Zeitraum nur wenige Aufzeichnungen gemacht, und die 30-Tage Testversion war ausgelaufen. Glücklicherweise, wurde bei Screencast-o-matic die zeitliche Beschränkung entfernt und man konnte so lange Videos aufnehmen wie man wollte. Das war auch das engültige Tool mit dem aufgeizichnet wurde.

3.3 Interviews und Fragebogen

3.3.1 Leitfaden der Interviews

Awareness ist ein subjektiv erlebtes Phänomen, daher wäre eine Analyse nur der äusseren Manifestation dessen, ohne der Schilderung der Erfahrun-

gen der Teilnehmer der Studie, unvollständig. Diese wurden mittels semi-strukturierten Interviews erhoben. Die Idee bei diesen ist, den Befragten ausreden zu lassen über die Erfahrung mit Saros und die Fragen eher als Leitlinien zu benutzen, mit denen man das Interview lenkt, als feste konkrete Fragen die man durchgehen muss. Die Fragen habe ich als losen Leitfaden für das Interview befolgt, damit wollte ich den freien Ausdruck damit nicht einengen und habe an vielen Stellen zu diesem Zweck frei geredet, meine eigenen gleichartigen Erfahrungen mitgeteilt und mich darum bemüht aus dem Interview ein ungezwungenes Gespräch zu machen. Die interessanten Aussage habe ich stichwortartig notiert, und nach den Interviews aus meinem Gedächtnis rekonstruiert.
Die Fragen sind im Anhang A zu sehen.

3.3.2 Fragebogen

Im Laufe der Analyse wurde klar, dass es WAMs in Saros gibt, die als solche in den Aufzeichnungen keine sichtbare Rolle spielen, da sie nur passiv genutzt werden. Über diese war ich nicht im Stande eine Aussage zu treffen, es war nicht zu sehen ob die Teilnehmer der Sitzungen überhaupt über dessen Existenz wissen. Diese sind solche wie das Kennzeichnen der offenen Dateien im Package Explorer, welche sich der andere Teilnehmer zur Orientierung anschauen kann, oder das Lesen der Markierungen des anderen Teilnehmers, so wie die Viewports und die Anzeige in der Rosterview welche angibt ob der Teilnehmer sein Eclipse-Fenster nicht im Fokus hat. Zum Zweck mir ein genaueres Bild um die passiven WAMs zu gewinnen habe ich einen Fragebogen zusammengestellt und beantworten lassen.
Im Fragebogen wurde kurz dessen Sinn erläutert, die einzelnen Workspace Awareness Mechanismen nochmal vorgestellt, und zu jedem WAM die Frage gestellt ob dieses Feature bei Erfüllung der Aufgabe geholfen hat, die Antwortmöglichkeiten: “häufig“, “gelegentlich“, “selten“, “nie“, “es hat gestört“ und ”was ist das?”. Es wurde erwartet das man eine Aussage darüber machen kann welche WAMs besonders nützlich sind, welche nicht nützlich, welche störend und welche unterrepräsentiert sind.
Dabei wäre interessant bei den unterrepräsentierten noch den Gesichtspunkt ob diese denn für die Teilnehmer relevant gewesen sein könnten, dies wurde aber aus Kostengründen nicht gemacht. Dabei hatte jeder die Möglichkeit die eigene Antwort zu kommentieren um besondere Umstände zu hervorzuheben. Ich habe auch die Gelegenheit genutzt um am Ende noch ein Paar frei zu beantwortende Frage zu stellen, zu dem was im Sinne der WAM besonders gut funktioniert, und was zu beanstanden war.

3.4 Die Grounded Theory

3.4.1 Fragestellung

Eine Technik der GTM, um aus einer möglicherweise Unmenge an Daten an die interessanten Daten dran zu kommen, ist die Fragestellung. Was die ausmacht ist folgendes, man formuliert eine Fragestellung mit der man an die Daten rangeht. Man stellt sich die formulierten Fragen ständig über die Daten, und zwar während des gesamten Prozesses der Analyse. Dies ermöglicht den interessanten Aspekt hervorzuheben und ist ebenso ein Mittel um in nicht in "der Flut der Daten zu ertrinken" [SS07]
Meine Fragen waren

- Werden jetzt relevante Workspace Awareness Informationen von Saros übertragen?
- Oder werden die nicht übertragen es wird eine Alternative benutzt?
- Werden Awareness Information übertragen bzw. nicht übertragen, dessen Relevanz nicht einzuschätzen ist?

Dabei habe ich die letzte Frage nur in Fällen wo ich eine Vermutung hatte, dass diese relevant sein könnte und gegebenfalls später im kontinuierlichen Vergleich eine Rolle spielen würde. Hätte ich alle Ereignisse die unter diese Kategorie fallen kodiert, wäre der Rahmen dieser Arbeit gesprengt.

Die Frage der Relevanz ist für die Kodes entscheidend gewesen. Da die GTM schreibt vor, dass alle Hypothesen in den Daten verankert sein sollen, waren für mich die Kriterien für die Relevanz nur die die explizit zu beobachten sind. Das heißt nur die Ereignisse wenn ein Entwickler, den Zustand seines Workspaces verbalisiert hat ("Ich bin jetzt im Follow-Mode"), oder nach dem Zustand des Workspaces gefragt ("Wo bist du gerade¿', "In welche Datei bist du?"). Es gab natürlich Ausnahmen, wie wenn zum Beispiel ein Entwickler etwas explizit markiert hat und dies Teil seiner Erklärung war, die Markierung aber nicht übertragen wurde, da sie sich mit einem Edit desselben Entwicklers überlappt hat [2], oder wenn jemand was erklärt hat ohne sich bewusst zu sein, dass der andere gerade wartet dass ein Programm mit der Ausführung endet, was ihn in der Mitverfolgung der Erklärung im Code hinderte. Die Relevanz des letzten Beispieles ist in letzten Fall natürlich offen zur Diskussion, da keiner hier über den Zustand, dass der eine Entwickler gerade jetzt blockiert ist oder ob er mitverfolgen kann, sich geäussert hat.

Im Rahmen dieser Arbeit wurden Ansätze der Grounded Theory verwendet, Die Kodes hab ich so ausgesucht das sie möglichst atomar und differenziert sind, so dass im Nachhinein ein reicherer Vergleich gestellt werden kann.

[2] dazu habe ich den Bug Report 3390589 im Tracker von Saros erstellt

Das hat als Konsequenz, das manche der Kodes eher deskriptiv als abstrakt sind, was im Gegensatz zu dem GTM-Ansatz steht [GG96].
Ein Objekt ist ein Gegenstand auf dem sich die Tätigkeit bezieht, dies ist manchmal abstrakt wie Position bis konkret wie Javadoc sein, das Kriterium bei der Namensvergabe war dass dies spezifisch genung ist für die noch Manchmal ist das Objekt sehr spezifisch benannt worden, wie zum Beispiel bei *ReportOnImplementingClassesView*.
Beim gesamt Überblick über die Konzepte wurde ersichtlich, dass die Konzepte alle Hindeutungen auf ein Problem in der Übertragung von Awareness Informationen sind. Nach dem ich die Konzepte gruppiert habe, bei manchen nach dem Objekt auf welches die sich beziehen, und bei anderen auf die Art der Tätigkeit die bezeichnen, sind daraus folgende Kategorien entstanden.

3.4.2 Die erarbeiteten Kategorien und Konzepte

Coordination Die hier enthaltenen Kategorien bezeichnen Konzepte die verschiedene Bereiche der WA Kennzeichnen, in denen sich WA als wichtig erwiesen hat. Alle Konzepte dieser Kategorien umfassen Aktivitäten der Teilnehmer, dessen Ziel war die Koordination untereinander war. Hier wurden unter anderem Informationen über den Zustand des Workspaces, dem Verlauf der Aufbau der Session, sowie Berichte darüber ob einer gerade verhindert ist in der Sitzung weiter zu machen, auf Grund des Verhaltens vom Eclipse.

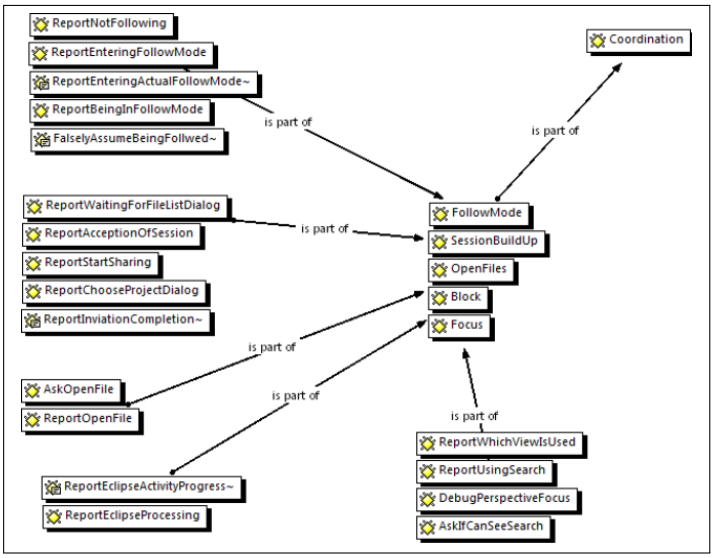


Abbildung 7: Die Kategorie *Coordination*, samt den dazugehörigen Konzepten

FollowMode Diese Kategorie bezeichnet Vorfälle in denen gefragt wird ob der Buddy gerade in Follow-Mode ist, oder der Buddy berichtet eigenständig darüber, dass er in Follow-Mode übergangen ist, oder sich bereits in ihm befindet.

SessionBuildUp Hier sind alle möglichen Berichte und Fragen die geäussert wurden während dem Aufbau der Session und der anschliessenden Synchronization. Typisch waren Aussagen wie "So, ich habe jetzt deine Einladung bekommen", "Bei mir läuft die Synchronisation noch" etc. Alle Aussagen bezogen sich auf verschiedene Stadien des Aufbaus.

OpenFile Hier handelt es sich um Fragen und Berichte die über die Position in einem Projekt ("Welche Datei?") oder innerhalb einer Datei berichten.

Focus Etwas ähnlich zu *OpenFile*, es handelt sich jedoch darum welche View oder Hilfestellung (wie der Content-Assistant[3]) bedient wird. Dabei wurden Informationen darüber ausgetauscht, welche der Views oder Hilfestellungen gerade bedient wird, worauf die Aufmerksamkeit des anderen Entwicklers gerade gerichtet ist.

Block Hier wurde über Blockierungen berichtet, welches den Fluß der Session beeinträchtigt haben. Dies ist zum Beispiel häufig eine Hintergrundsoperation von Eclipse gewesen.

TaskExecution Im Unterschied zu Coordination sind hier Aktivitäten der Teilnehmer erfasst worden, bei denen die vermittelten WAIs unmittelbar zur Erfüllung der gegebenen Aufgabe dienten.

AidedExplanation Wenn ein Entwickler irgendwas erklärt und sich dabei eines zusätzlichen Werkzeugs bedient hat, wie zum Beispiel den Markierungen oder einer View, wie die Konsole oder dem Eclipse Navigator[4].

ExplicitSync Diese Kategorie umfasst all die Konzepte bei denen die betroffenen Workspace Elemente explizit synchronisiert worden sind. Was das genauer heißt ist das der Driver ein Werkzeug bedient und dabei etwas erklärt, der Observer dabei dasselbe Werkzeug bedient und beinahe dieselben Manipulationen an diesen vornimmt, ohne dabei das diese explizit kommuniziert werden, sondern der Observer die dazu benötigte Information aus dem Kontext der Erklärung entnimmt .

[3] ein Hilfswerkzeug, welches für getippten Code programmierspezfische Hilfestellungen bietet

[4] Der Navigator ist eine View, die die Verzeichnisstrukturen in Eclipse darstellt

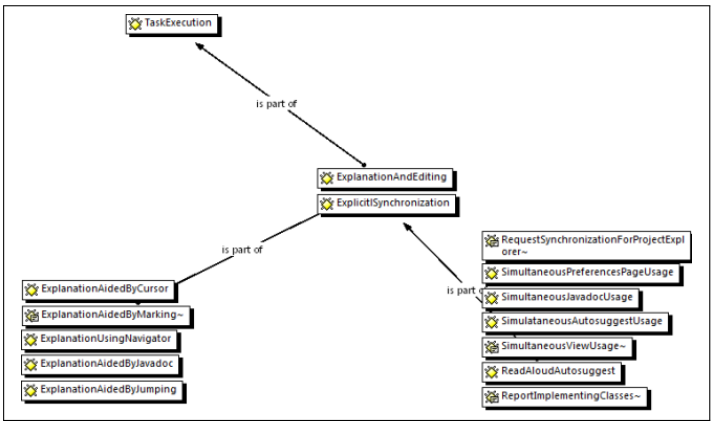


Abbildung 8: Die Kategorie *TaskExecution*, samt den dazugehörigen Konzepten

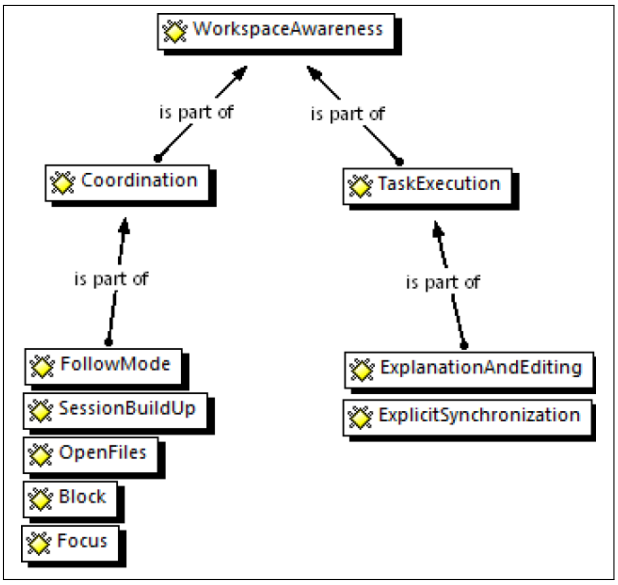


Abbildung 9: Die entstandenen Kategorien, zusammengefasst

3.4.3 Die Auswertung

Im vorherigem Abschnitt haben wir die Konzepte und Kategorien eingeführt die es helfen über die WAMs in Saros nachzudenken. Nun können wir zum eigentlich interessanten Teil dieser Arbeit kommen, zu der Auswertung. Hier werden die Konzepte und Kategorien bezüglich dessen ob sie übertragen

worden sind oder nicht untersucht, und es wird auf interessante Fälle genauer eingegangen. Ferner werden hier Aussagen der Teilnehmer aus den semi-strukturierten Interviews miteinbezogen, und wo angemessen untermauere ich meine Aussagen mit den Ergebnissen des Fragebogens.

3.4.4 Relevante Informationen die übertragen werden, wie gut?

Hier wird auf die einzelnen WAMs eingegangen Aus den Vorkomnissen die aus der Analyse der Videos gekommen ist der Kategorie *AidedExplanation*, wie sie beschrieben worden ist, aus den Interviews und dem Fragebogen kam zu Vorschein, dass das Markieren für die Erfüllunf der Aufgabe besonders hilfreich gewesen ist, wie man in 10 sehen kann. Häufig wurde in den Interviews und in dem Fragebogen erwähnt auch dass das gleichzeitige Editieren besonderes hilfreich ist. Seltener benutzt aber trotzdem positiv bewertet ist der markierte Zeiger gewesen, den aber nicht alle bemerkt haben. Einer der Teilnehmer hat sich geäussert, das er auf diesen verzichtet hat und sich lieber den Viewports und der Markierungen bedient hat. Anhand dessen ist keine eindeutige Aussage über den Zeiger zu treffen, obwohl er unterrepräsentiert ist, wird er benutzt, aber seine Aufgabe kann aber wohl mit anderen Mitteln erledigt werden.

Nach dem Fragebogen haben die meisten die Anzeige der offenen Dateien im PackageExplorer selten oder gar nicht benutzt, die Mehrheit hat diesen Mechanismus überhaupt nicht bemerkt. Hier ist nochmals zu erwähnen, dass Saros auch die Dateien die im Hintergrund offen sind im PackageExplorer anzeigt, und dass diese Informationen in den Aufzeichnungen nicht zum Vorschein gekommen sind. Die Vorfälle aus der Kategorie *OpenFiles*, die Fragen und Berichte darüber enthält darüber welche Datei der andere gerade offen hat,ist ein Hinweis drauf, dass diese wichtig sind. All dies lässt drauf schliessen, dass dieses WAIs wichtig sind, öfters ist der Mechanismus selbst nicht bekannt gewesen.
Über die Viewports, welche im Editor den Bereich markieren welchen der andere Teilnehmer sieht, kann man nur eine Aussage aus dem Fragebogen entziehen, da diese in den Aufzeichnungen nicht zum Vorschein gekommen sind. Die sind gelegentlich genutzt worden, obwohl manche diese gar nicht bemerkt haben. Manche haben es statt dem Zeiger benutzt um die Position des Anderen innerhalb der Datei zu erkunden. Obwohl dieses Feature etwas unterrepräsentiert ist, hat es sich als durchaus nützlich erwiesen um die Sicht des anderen darzustellen.

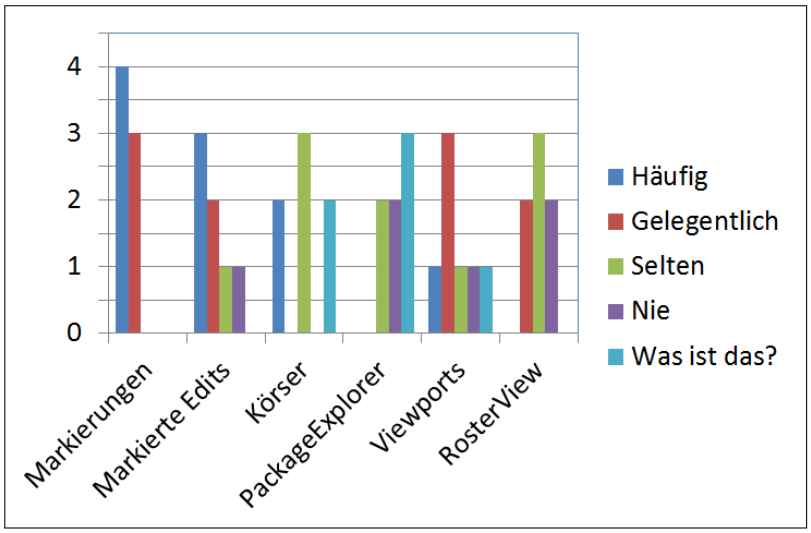


Abbildung 10: Die Ergebnisse d. Fragebogens, die y-Achse ist die Anzahl d. Teilnehmer die die Antwort angekreuzt haben

3.4.5 Relevante Informationen die nicht übertragen werden

In diesem Abschnit werden die Kategorien erläutert, die auf einen Mangel in der Übertragung der WAI hinweisen. Fast alle Kategorien aus *Coordination* fallen hier mit rein, so auch aus *ExplicitSync* und *TaskExecution*.

Die Konzepte die unter *FollowMode* klassifiziert worden sind, deuten auf einen Mangel von Awareness beim Driver, darüber ob Seine Sicht vom Observer gerade befolgt wird. Die Teilnehmer haben sich häufig darüber geäussert ob sie angefangen haben zu folgen, ob sie nun folgen, oder dass sie nicht folgen. Einmal gab es den Fall wo ein Entwickler im Follow-Mode war, zwischendurch aber eine andere Datei geöffnet hatte, und dadurch aus dem Follow-Mode rausgekommen war, ohne sich dessen bewusst zu werden. Er begann daraufhin über eine bestimmte Code-Stelle zu reden und in dieser explizit zu markieren, dabei hatte sich der Andere bereits mit einer anderen Stelle im Code angefangen zu befassen. Dass es ihm nicht bewusst war schließe ich aus seiner anschließenden Aussage "Ich bin ja gar nicht in Follow-mode.". Dies lässt auf zwei Probleme hindeuten, zum einen sind Informationen darüber ob ein Teilnehmer folgt mangelhaft und im oben beschriebenen Fall ist zusätzlich noch das Verhalten des Follow-Modes dem Entwickler unklar gewesen.
Was in den Interviews ausdrücklich als negativ bewertet wurde, war der langwierige Aufbau der Session, bei der man über längere Zeitstrecken kein Feedback darüber bekam was beim anderen gerade geschieht und, was gerade Saros "im Hintergrund macht". Der letzte Aussage bezieht sich darauf, dass

nach dem man eine Einladung verschickt wird, Saros aktuell keinen Feedback gibt, so dass der Entwickler nicht weiss ob es sich gerade um eine Fehler handelt oder ob Saros Hintergrundspoerationen gerade ausführt. Dabei haben sich die Entwickler darüber informiert ob eine Einladung angekommen ist und akkzeptiert worden ist, welche Dialoge angezeigt werden, wie der Zustand des Ladebalkens ist ("Es ist gerade bei 76%") und wann der Einladungsprozess samt Synchronisierung abegschlossen ist("Yoohoo!"). Es ist natürlich fraglich ob die hier erwähnten Beobachtungen tatsächlich ein Problem der Awareness darstelen, weil alle diese Informationen kennzeichnen auch einfach ein mangelndes Feedback von Saros während dem Aufbau der Session. Es liegt mir nahe zu vermuten, dass diese Informationen als wichtig angesehen wurden, weil diese die einzigen Anhaltspunkte über den Fortschritt des Aufbaus der Sitzung sind.
Die verschiedenen Views und Hilfestellung die Eclipse bietet werden in Saros nicht alle unterstüzt, sind aber für die Entwicklung wichtig, worauf die Vorfälle aus *ExplicitSync* hindeuten. Dabei hat der Driver ein Werkzeug bedient und dabei was erklärt, der Observer dabei dasselbe Werkzeug bedient und beinahe dieselben Manipulationen an diesen vorgenommen, ohne dabei ,dass diese explizit kommuniziert werden, sondern der Observer entnahm die dazu benötigten Informationen wahrscheinlich aus dem Kontext der Erklärung. In anderen Fällen wurde aus den Views und Hilfestellungen einfach vorgelesen. Einmal wurde sogar gefragt danach, dass der Driver eine Datei öffnet so das sich der Observer wieder in dem damals benutzten ProjectExplorer orientieren kann ("Kannst du mal eine Datei öffnen, damit ich weiß wo du bist?"). Weiterhin wurden auch Inhalte der Views oder Hilfestellungen vorgelesen. All dies deutet darauf hin, dass Informationen über diese Elemente in Saros nicht im optimalen Umfang dargestellt werden.

4 Lessons learned über Datenerhebung im industriellen Betrieb

In diesem Kapitel möchte ich auf meine Erfahrung bei der Datenerhebung in einem industriellen Umfeld eingehen und die Probleme, auf die ich dabei gestoßen bin sowie deren Lösungen.

Zunächst ein Paar Worte über mein Verhältnis zu der Firma in der die Aufzeichnungen gemacht worden sind. Ich bin bei dieser seit zwei Jahren tätig. Angefangen habe ich als Praktikant in der IT, und nach dem Ende des Praktikums habe ich dort eine Stelle als studentische Aushilfskraft angenommen. Ich war intensiv an dem Entwickungsprozess beteiligt und habe kleine bis mittelgroße Aufgaben übernommen. Zu dem Betrieb kann ich sagen, dass obwohl Zeitdruck immer da ist, eine freundliche und solidarische Atmosphäre herrscht.

Am Anfang dieser Arbeit war mein geplantes Vorgehen für die Datenerhebung, eine Präsentation über Saros zu machen und einen HowTo (siehe Anhang D) zu verschicken der die Installation und die Benutzung von Saros erläutert. Saros wollte ich nach und nach einführen. So wie die Aufgaben kamen wollte ich den Entwicklern die Software am Arbeitsplatz kurz nochmals vorstellen. Als Aufgaben haben wir uns auf solche geeinigt, bei denen es "passt", d.h. dies in eins der vier Szenarien für die Benutzung von Saros passen.

In der Tat hat dieser Plan zur großen Frustration, nicht funktioniert. Zum einen gab es das Problem von Zeitdruck und den Mangel an geeigneten Aufgaben für Saros, sodass man manchmal mit etwas Druck nachhelfen musste, damit eine Aufzeichnung gemacht wird. Ich erwähne hier nochmals, dass PP in dem Betrieb bis dato nur im Einzelfall praktiziert worden war. Das hatte als Auswirkung, dass die Entwickler nicht sagen konnten "Das ist eine Aufgabe die wir normalerweise mit PP machen würden, hier können wir aber auch mit Saros arbeiten".

Der Schlüsselmoment war der Vorfall, als wieder eine Aufzeichnung nicht aufgenommen wurde, weil die geplante Aufgabe sich von den Anforderungen verändert hatte, und die beiden Entwicklern einfach unabhängig von einander gearbeitet haben. Hier entschied ich mich dafür eine Mail an den Leiter der Entwicklung zu schreiben, in der ich meine Enttäuschung über den Verlauf der Aufzeichnungen geäußert habe. Dabei habe ich den Vorstandsmitglied auf CC gesetzt, das die Unterstützung für diese Arbeit begünstigt

hat. Prompt kam die Antwort, dass ab dem nächsten Sprint[5](welcher zwei Wochen dauert) Aufgaben gewählt werden, die mit Saros erledigt werden müssen. So wurde die Vorgabe mit Saros zu arbeiten unmittelbar von der Leitung gegeben, und die Aufgaben wurden von dem Team selbst bestimmt, dadurch war die Verbindlichkeit gestiegen. Nach dieser Vereinbarung wurden mehr Aufzeichnungen gemacht, aber leider nicht in der gewünschten Anzahl.

Vor dem Anfang des nächsten Sprints habe ich nochmal mit dem Leiter der Entwicklung gesprochen und berichtet, dass die Anzahl der Aufzeichnungen immer noch für eine vernünftige Analyse nicht ausreicht. Darauf hat er dem Team erklärt, dass im nächsten Sprint jede Aufgabe mit Saros zu erledigen sei. Dabei ist man unglücklicherweise auf das Problem gestoßen, dass die Testlizenzen für die Aufzeichnungssoftware, Camtasia Studio 7, bei den meisten Entwicklern ausgelaufen waren. Das Umstellen auf eine andere Software hat natürlich ein wenig Zeit in Anspruch genommen.
Es wurde mir klar, dass es einen Sprint, in dem jede Aufgabe mit Saros aufgezeichnet wird, aus Kostengründen nicht geben wird. Daher entschied ich mich für die Saros-Sitzungen feste Termine am Anfang des Sprints zu vereinbarens. Die Termine habe ich einer Liste zusammengetragen und diese Liste an einem zentralen Ort hinterlegt(in diesem Fall bei der Kaffeemaschine) (die Liste ist im Anhang E zu sehen).

Allgemein habe ich für mich folgende Schlüsse gezogen:

1. Alles einrichten (Installationen + Accounts), so dass die Entwickler sofort los starten können, und ihre Zeit nicht verschwenden.

2. Aufzeichnungen verbindlich machen, da man in einem Betrieb, bei dem es Zeitdruck gibt, nicht von den Entwicklern erwarten kann, dass sie dafür Eigeninitiative aufbringen.

3. Bei Problemen, sofort mit der Leitung reden, da diese die Autorität besitzt um die nötige Kooperation herbeizuführen.

4. Physisch vor Ort sein während einer Aufzeichnung um eventuell bei Problemen zu unterstützen, da die Entwickler lieber erstmal selbst versuchen werden das Problem zu lösen, und sich erst im kritischsten Fall melden werden.

5. Aufzeichnungen mit einer Software machen und stark dagegen steuern, dass eine andere verwendet wird, da wenn diese versagt, dies die ganze Aufzeichnung unbrauchbar machen kann.

.

[5]eine Iteration des Entwicklungsprozesses in Scrum, an dessen Anfang die im Sprint zu erledigenden Aufgaben bestimmt werden

5 Unterstützung im Rahmen des Saros Entwicklungsteams

Im Laufe der Bachelorarbeit war ich für die verfügbare Zeit Teil des Saros Entwicklungsteams, obwohl ich an der eigentlichen Entwicklung, im Sinne von Quellcode schreiben, nicht beteiligt war. Hier muss erwähnt werden, das meine Anwesenheit nicht so war wie bei den anderen *Sarosianer*, da ich häufiger bei der Firma sein musste, um bei den Aufzeichnungen Unterstützung zu leisten. Während dieser Zeit habe ich den Entwicklungsprozess von Saros kennengelernt, an den Stand-Up Meetings teilgenommen so wie meine Rolle im monatlichen Releasezyklus von Saros erfüllt.
Im Saros-Entwicklungsteam, werden jeden Dienstag und Donnerstag Stand-Up Meetings gehalten. In diesen Meetings wurde diskutiert was die *Sarosianer* seit dem letzten Meeting gemacht haben, welche Probleme es dabei gab und was sie bis zum nächsten Meeting tun werden. Was für mich besonders hilfreich war ist der Wissenstransfer der hier angestoßen wird. So hatte man die feste Möglichkeit sich über die eigene Arbeit auszutauschen, und bei Problemen Hilfe von den erfahreneren Mitglieder der Arbeitsgruppe zu bekommen. So könnten bei mir manche Probleme früh identifiziert, und neue Ideen miteingebracht werden.
In der letzten Arbeitswoche jeden Monats wird ein neues Saros-Release rausgebracht. Um das Release zu liefern und dessen Qualität zu sichern werden verschiedene Aufgaben erfüllt. Für jede dieser Aufgabe ist eine Rolle zuständig.
In meiner Zeit erfüllte ich einmal die Rolle des Test-Managers(TM) und einmal die des TM-Assistenten (TMA). Der Test-Manager ist dafür zuständig, dass die neuen Features und Fixes getestet werden, verifizierte Fixes geschlossen und noch kritische Bugs gemeldet werden. Vor dem Liefern des Releases führt der TM einen Sanity-Test durch und lässt die automatisierten Tests durchlaufen. Bei allen Aufgaben wird der TM von dem TMA auf unterschiedlichen Weisen unterstützt, wobei die wichtigste davon die Einführung in seine Rolle ist.
In der Regel war der TM ein Neuling, und der TMA jemand der schon einmal TM war. Durch den Umstand, dass der TMA das Wissen für das Test-Management hat, und der TM das Wissen nicht, aber die Verantwortung, findet auf eine natürliche Art und Weise Wissenstransfer statt.
Ich hatte das Glück, dass mein TMA Karl Held war, der mich sehr eingängig in diese Rolle eingeführt hat, und dass das Release als solches keine kritische Probleme hatte. Zusammen mit Karl Held haben wir für die neuen Features und Fixes manuelle Tests erstellt, und dabei gegebenfalls Rücksprache mit den jeweiligen Entwickler gehalten. Am Tag vor dem Release haben ich und Karl Beecher den Sanity-Test gemacht und die automatisierten Test ausgeführt. Alles in allem ist der ganze Prozess reibungslos verlaufen.

Beim nächsten Release sollte ich für den TM Hendrik Degener als TMA mitwirken. Ich habe ihm eine Einführung in die Rolle gegeben und Einzelheiten bezüglich des Schreiben der Tests diskutiert. Leider konnte ich diese Rolle wegen meines Zeitplanes nicht völlig ausfüllen, und musste früher die Sitzung verlassen.
So gering wie meine Beteiligung auch gewesen ist, fand ich diese Erfahrung lehrreich und angenehm.

Im Laufe meiner Analyse, sind die Teilnehmer der Feldstudie auf Probleme gestoßen. Ich habe diese notiert und in den Tracker von Saros eingepflegt. Das sind folgende gewesen:

3393918	Eclipse froze when screensharing
3397399	Notification ”Buddy joined the session”counterintuitive
3390589	Highlighting not specific enough

So wie einen FeatureRequest:

3401090	Add ”followers”to driver in roster

6 Fazit

Das Ziel dieser Arbeit war die Evaluation der WAMs in Saros. Diese habe ich mit einer qualitativen Analyse von Aufzeichnungen von Saros-Sitzungen gemacht. Die qualitative Analyse habe ich mit Ansätzen aus der GTM vorgenommen. Zum Zweck der Datenerhebung habe ich zweierlei Aufgaben erfüllt. Dafür habe ich zum einem eine Aufzeichnungsinfrastruktur aufgesetzt. Dabei habe ich nach der Software gesucht, die für die Aufzeichnung und die anschließenden Bearbeitung der Videos am besten geeignet wäre. Zum anderen, habe ich nach und nach Saros in der Firma wo ich eingestellt bin eingeführt. Dabei musste ich lernen wie schwer dies in einem laufenden industriellen Betrieb sein kann, und wie man damit umgeht.
Für die qualitative Analyse, habe ich mich mit der GTM auseinander gesetzt. Ich habe dem Grundgedanken der GT, dass alle entstehenden Hypothesen in den Daten verankert werden sollen, stets versucht zu folgen. Dabei habe ich die Techniken die die GTM vorgibt, in einem nicht striktem, aber für Zwecke dieser Arbeit angemessenem, Maße praktiziert. Diesen Ansätzen folgend, habe ich Videos, semi-strukturierte Interviews und Fragebögen analysiert, und bezüglich der Übertragung der Awareness ausgewertet.
Dabei bin ich zu folgenden Schlüssen gekommen: alles in allem kann man sagen das Saros in Hinsicht auf die Kategorie *TaskExecution* die das gemeinsame Editieren von Quellcode erlauben, die Entwickler in einem zufriedenstellendem Maße unterstüzt. Zur Task-Ausführung haben sich die Teilnehmer wiederholt gewünscht, dass man die Views und den Debug-Modus synchronisieren könnte, was man mit Screensharing erreichen kann, wobei dieses Feature leider zum Zeitpunkt der Untersuchung nicht benutzbar war. Die WAMs die bereits vorhanden sind, und die Koordination in einem Projekt unterstützen sollten sind unterrepräsentiert, so dass sich manche Teilnehmer explizit über die offenen Dateien oder über die gerade verwendeten Views und Hilfestellungen austauschen müssen .
Der Follow-Mode funktionert gut als solcher, obwohl es dem Driver an Feedback darüber mangelt wer ihm gerade folgt. Gelegentlich war das Verlassen des Follow-Mode nicht transparent.
Nach Aussagen der Teilnehmer der Feldstudie ist der größte Schwachpunkt der Mangel an Feedback während des Aufbaus der Session. Dieser kann eventuell als ein Awareness Problem erfasst werden, muss es aber nicht.

7 Ausblick

Was leider in der Analyse nicht untersucht werden konnte, ist das Screensharing Feature von Saros. Es wäre nötig die Untersuchung nochmals zu machen, um zu sehen welche von den erwähnten Problemen durch das Benutzen dieses Features wegfallen würden, und welche Probleme vielleicht neu

dazu kommen würden.

Es wäre auch interessant dieselbe Arbeit auf einem breiteren Datensatz zu machen, um eine statistische Auswertung der WAMs in Saros zu machen.

Ferner könnte sich eine andere Arbeit mit dem Erarbeiten eines Codiersschemas für die Workspace Awareness befassen, da das hier verwendete Schema sich auf einem ziemlich deskriptivem Niveau sich befindet und für Zwecke der Evaluation gemacht worden ist. Um ein allgemeingültigeres und handhabareres Schema zu gewinnen, müsste die Analyse über mehrere Iteration durchgehen. Ich sehe hier durchaus Potential, wenn man einen genaueren Codierblickwinkel einnimmt und die Beziehungen zwischen den Konzepten noch feiner ausarbeitet.

Ferner, wie aus der Auswertung ersichtlich geworden ist, gibt es Bereiche in Saros wie den Follow-Mode und die Einladung in denen die WAMs noch weiterentwickelt werden müssen. Andere Mechanismen die unterrepräsentiert sind, wie zum Beispiel der PackageManager, könnten so überarbeitet werden, dass sie mehr zum Vorschein kommen.

Ferner könnten WAMs in das bestehende Tutorial eingepflegt werden, so dass bei der ersten Benutzung, ein Nutzer auf die etwas unaufälligeren WAMs aufmerksam gemacht wird.

A Leitfragen für die Interviews

- Wie war allgemein die Erfahrung mit dem Tool?
- Gab es was besondere Erfahrungen die man mitteilen möchte?
- Gab es Punkte wo Awareness Informationen sehr wichtig waren?
- welche waren das?
- Gab es Punkte wo einem Awareness Informationen gefehlt haben?
- Welche? Was hätte man sich da gewünscht?
- Gab's auch Punkte wo einer den Eindruck hatte, die Awareness Informationen wären besonders geschickt übertragen?

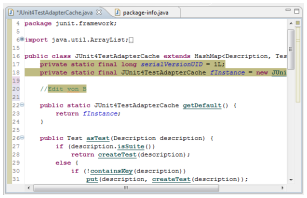
B Fragebogen

Fragebogen bestehende Präsenzverfahren in Saros

Eine der Definitionen für Workspace Awareness lautet *„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“*

Saros ist mit zahlreichen Präsenzverfahren ausgestattet, die die Workspace Awareness während einer Session darstellen. Ziel dieses Fragebogens ist meine Evaluation zu vervollständigen insbesondere um die Präsenzverfahren die passiv benutzt werden und aus den Videos es nicht ersichtlich ist, wie diese zu evaluieren sind. Als Beispiel, wenn man ein Stück Code erklärt und dabei sich mit Markierungen hilft, ist dies aktiv, wenn man aber diese Erklärung wahrnimmt wird das Feature passiv benutzt, oder wenn man die Edits des anderen sieht und dies einem in der Orientierung hilft.
Hier werden aber auch allgemeinere Fragen gestellt. Es wäre auch toll, wenn ihr eine kurze Begründung angibt, zum Beispiel wenn man nur Codeeinführungen gemacht hat, kann man hier kaum eine Aussage machen.

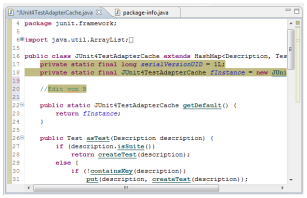
Markierungen



Dieses Feature hat mir bei der Erfüllung der Aufgabe geholfen:

Häufig Gelegentlich Selten Nie Es hat gestört Was ist das?

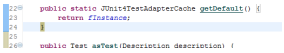
Markierte Edits das was der andere Buddy geschrieben hat, wird ebenso farblich markiert



Dieses Feature hat mir bei der Erfüllung der Aufgabe geholfen:

Häufig Gelegentlich Selten Nie Es hat gestört Was ist das?

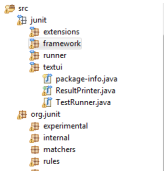
Der **Körser** wird mit der kleinen Markierung in Zeile 24 dargestellt



Dieses Feature hat mir bei der Erfüllung der Aufgabe geholfen:

Häufig Gelegentlich Selten Nie Es hat gestört Was ist das?

Package Explorer ein gelber Punkt Decorator zeigt an ob der Buddy die Datei offen hat, ein grüner, dass er diese gerade im Fokus hat, der blaue Doppelpfeil ob die Datei synchronisiert ist.



Dieses Feature hat mir bei der Erfüllung der Aufgabe geholfen:

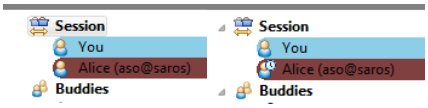
Häufig Gelegentlich Selten Nie Es hat gestört Was ist das?

Viewports geben an welchen Bereich der Buddy vom sichtbaren Bereich nun sieht (links), und welchen Bereich der Buddy von der ganzen Datei sieht (rechts)

Dieses Feature hat mir bei der Erfüllung der Aufgabe geholfen:

Häufig Gelegentlich Selten Nie Es hat gestört Was ist das?

Rosterview zeigt die Buddyliste und die Teilnehmer der Session an, zusätzlich wenn ein Buddy gerade ein anderes Fenster als Eclipse im Vordergrund ist, wird eine Uhr bei dem Buddy angezeigt



Dieses Feature hat mir bei der Erfüllung der Aufgabe geholfen:

Häufig Gelegentlich Selten Nie Es hat gestört Was ist das?

Hier bitte mit einem kurzen Satz antworten:

- Ich fand folgende Feature besonders hilfreich:
- Mir haben während der Session folgende Informationen über die Tätigkeiten des Anderen gefehlt:
- Ich möchte noch dazu sagen, dass:

C Präsentation

andrä ag

Saros

- Eclipse PlugIn für DPP und kollaboratives Editieren
- „Buddies“ arbeiten in einer Session auf einer synchroniserender Kopie eines Projektes
- Awareness Information: Tätigkeit und Sicht d. Anderen wird farblich markiert
- Sessions über XMPP/Jabber Server
- VoIP

andrä ag

Saros

- „Synchroniseren“
  - Der Einladende teilt ein Projekt mit seinen Buddies mit, das Projekt wird übertragen, eventuelle Unterschiede werden überschrieben

andrä ag

Aufgabenarten

- Gemeinsames Review:
  - 1 Driver der den Code durchschaut
  - mehrere Observer in „follow mode“
- Einführungen
  - dasselbe wie davor, mit dem Vorteil das ein Observer für sich Stücke Code anschauen kann, ohne die Session zu beeinträchtigen

andrä ag

Aufgabenarten

- Distributed Party Programming
  - 2 bis 5 Beteiligten programmieren unabhängig, an einer identischen Kopie von dem Projekt. Bei Bedarf kann einer helfen, ohne den Sitz zu verlassen. Was auch Zeit spart, ist das Konflikte früh erkannt werden
- Distributed Pair Programming
  - Sonderfall von DPP, ausser dass hier 2 Entwickler/innen nahe zusammen arbeiten und die Besten ihrer Ideen kombinieren

andrä ag

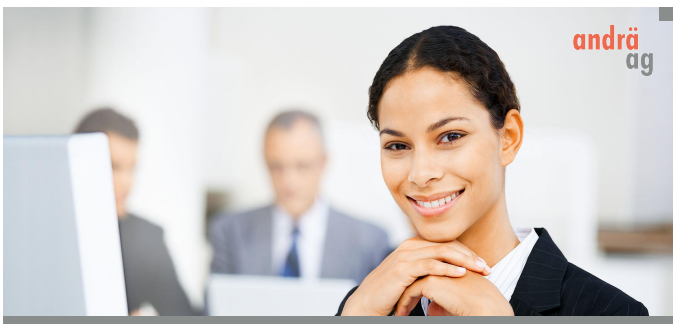
Wo es sich bei uns eignet

- Vorteil, dass man gleichzeitig programmieren kann
- Reviews und Einführungen
- Neue Features bzw. Fixes – wenn diese nicht zu trivial sind
  - Bei fehleranfälligen Kodierungstasks, bei zB komplexer Logik
  - Qualität von Code und Design wichtig, bei zB zentrale Komponenten oder kritische Features
  - Eine Übertragung von Wissen erwünscht ist, insbesondere bei Themen wo sich zur Zeit noch wenige auskennen

andrä ag

Aufzeichnung

- Teil meiner Bachelorarbeit ist es, Aufzeichnungen eurer Arbeit zu machen
- Headset
- Webcam
- Camtasia 7
- HowTo



Danke fürs Zuhören,
jetzt kommt die Demo!

D HowTo

Saros installieren

In Eclipse durch *Help -> Install New Software* mit diesem Update Site:
http://dpp.sourceforge.net/update

Als Jabber-Server bitte unseren hausinternen saros.a2gb.de nehmen. Um das Hinzufügen von Buddies weniger umständlich zu machen, nimmt bitte Eure Namenskürzel als Username. Für die Freelancer wären die Namenskürzel

Lennart Reuther – lre

Sören Beyer – sbe

Sergei Vasiliev – sva

Wenn ihr bereits einen Account habt dann könnt ihr unter *Saros->Create XMPP/Jabber Account…* Euch einen neuen Account anlegen. Bitte dann auch bei *Saros->Preferences* sicherstellen dass dies auch der aktive Account ist.

Der erste Schritt …

…ist die Einführung zu lesen ;) Die findet man unter http://www.saros-project.org/GettingStarted, und die ist ganz gut geschrieben. Was ihr mitnehmen solltet ist, wie man einen Buddy hinzufügt, wie man eine Session startet, und was die verschiedenen Modi sind (am wichtigsten der Follow-Mode), so wie ganz wichtig für die tatsächliche Zusammenarbeit: http://www.saros-project.org/GettingStarted#Staying_Aware_of_your_Fellow_Par.

Danach könnt ihr einen Buddy hinzuzufügen, und mit ihm die Arbeit an eurem Task anfangen.

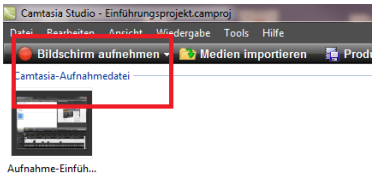
Headset + Webcam

Man sollte sich für die Aufzeichnung, beides besorgen. Wenn der Nachbar keinen Headset hat, kann man sich bei Arsenij oder Tobias melden. Zwei Webcams gibt es an Arsenij's Tisch

Camtasia 7

Die 30 Tage Testversion gibt es hier http://www.techsmith.de/download/camtasiatrial.asp

Nach der Installation Camtasia Studio öffnen, und auf Bildschirm aufnehmen klicken



Danach sollte folgendes Fenster aufpoppen

Da bitte die richtige Webcam und das Headset auswählen. Prüfen Sie nach, ob der aufzunehmende Bereich des Bildschirmes, das Eclipsefenster ist. Großen roten Knopf drücken. Wenn die Session vorbei ist auf den Stop Button drücken und unter Austausch:\aso\bachelorarbeit\aufzeichnungen\, als Name bitte den eigenen Namenskürzel, Unterstrich, Namenskürzel des Partners, Unterstrich, Datum nehmen. Zum Beispiel hwe_wru_0606.camrec. Das ist nötig damit ich später die Aufzeichnungen zueinander zuordnen kann. Und zum Schluss, bitte Aufzeichnung nicht pausieren, das ist danach sehr umständlich für mich die Videos zu schneiden.

Bitte nicht vergessen, die Rollen beim Pair-Progamming regelmäßig zu switchen ;) (siehe Follow-Mode im obigen Link)

Na dann, frohes Proggen ;)

E Liste

Saros Sessions diese Woche

Wer?	Welche Story?	Wann?	Done?

- So bald ein weiterer Termin bekannt wird, bitte eintragen.
- Falls Ihr in der Lage seid, eine Aufzeichnung länger als eine Stunde zu machen (2 oder 3), dann macht das bitte. Das hilft mir ungemein weiter
- Denkt dran, auf C:\ genügend Speicherplatz zu haben
- Bei besonderem Einsatz bei den Aufzeichnungen diese Woche gibt es je nach Präferenz Bier bzw. Kuchen ;)
    - Besonderer Einsatz heißt, dass ich glücklich werde
    - Ich werde glücklich, wenn eine Aufzeichnung
        - Lang,
        - Korrekt (keine kaputten Dateien, es sind beide Aufzeichnungen da, beide Dateien sind im selben Format)
        - Gut benannt ist (Namenskürzel + Datum müssen figurieren)

Literatur

[Car07] Jeffrey Carver. *The Use of Grounded Theory in Empirical Software Engineering.* Springer Verlag, 2007.

[Dje06] R. Djemili. Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierungverteilter Paarprogrammierung. Master's thesis, Freie Universität Berlin, Inst. für Informatik, 2006.

[GG96] Carl Gutwin and Saul Greenberg. *Workspace Awareness in Distributed Real-Time Groupware.* PhD thesis, University of Calgary, Department of Computer Science, 1996.

[SS07] Laura Plonka und Lutz Prechelt Stephan Salinger. A coding scheme development methodology using grounded theory for qualitative analysis of pair programming. In *Proceedings of the 19th Annual Workshop of the Psychology of Programming Interest Group*, 2007.

[SS10] Karl Beecher und Julia Schenk Stephan Salinger, Christian Oezbek. Saros: An eclipse plug-in for distributed party programming. In *CHASE '10 Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, 2010.

[uJC96] Anselm Strauss und Juliet Corbin. *Grounded Theory: Grundlagen Qualitativer Sozialforschung (Originaltitel: Basics of Qualitative Research).* Psychologie Verlags Union, 1996.