Freie Universität Berlin

Verteilte Paarprogrammierung im industriellen Umfeld
Etablierung und qualitative empirische Untersuchung

Diplomarbeit von
Edna Rosen

Eingereicht bei Prof. Dr. Lutz Prechelt

Betreuer: Christopher Oezbek und Stephan Salinger

25. Oktober 2009

Danksagung

Bei Herrn Prof. Dr. Lutz Prechelt bedanke ich mich für die Vergabe und Betreuung der Diplomarbeit.

Besonderen Dank schulde ich Christopher Oezbek und Stephan Salinger, die mich durch ihre engagierte Betreuung und stete Diskussionsbereitschaft mit vielseitigen Denkanstößen bereicherten und bei der Erstellung dieser Arbeit unterstützt haben.

Des Weiteren danke ich dem Saros-Team unter der Leitung von Christopher Oezbek und Stephan Salinger sowie allen anderen wissenschaftlichen Mitarbeitern für ihre vielen Hinweise und Hilfen während des Projektes.

Ich danke den Mitarbeitern von Teles, für ihre Mitarbeit und großzügige Unterstützung.

Nicht zuletzt möchte ich meinen Freunden und meiner Familie, besonders meinem Vater, danken, die mir durch ihre fortwährende Unterstützung das Studium und diese Arbeit ermöglichten.

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.

Berlin, 25. Oktober 2009

Edna Rosen

Kurzzusammenfassung

Durch die Globalisierung wuchs besonders in den letzten zehn Jahren der Bedarf, in international verteilten Teams gemeinsam Softwareprojekte zu verwirklichen. Paarprogrammierung ist eine agile Softwareentwicklungsmethode, die sich durch intensive Kommunikation und hohe Qualität in der Zusammenarbeit bereits in der Softwareentwicklung bewährt hat. Dabei erzeugen zwei Entwickler gemeinsam an einem Rechner Code. Dank neuester Technologien wie z.B. der in dieser Arbeit eingesetzten Groupware Saros ist es möglich, diese Methode der Softwareentwicklung auch in räumlich verteilten Teams einzusetzen.

Im Rahmen dieser Arbeit wird verteilte Paarprogrammierung als neue Softwareentwicklungsmethode in einem internationalen Unternehmen zur Unterstützung eines verteilten Teams von Entwicklern etabliert.
Die Etablierung wird mit Hilfe einer strukturierten Form der Handlungsforschung durchgeführt, die es erlaubt, optimierend in den Etablierungsprozess einzugreifen, und die es den Entwicklern ermöglicht, sich aktiv am Forschungsprozess zu beteiligen.

Diese Arbeit enthält eine Beschreibung der Ereignisse des Etablierungsprozesses sowie Erkenntnisse, die sich aus einer qualitativen empirischen Untersuchung möglicher Probleme von Etablierungsprozessen und des Nutzens von verteilter Paarprogrammierung im industriellen Umfeld ergeben haben. Des Weiteren wird aus den gesammelten Erkenntnissen eine allgemeine Anleitung für Etablierungsprozesse abgeleitet.

Die Ergebnisse zeigten, dass der Etablierungsprozess erfolgreich war und verteilte Paarprogrammierung trotz einiger Schwierigkeiten für die Zusammenarbeit von verteilten Teams über eine große Distanz nutzbringend eingesetzt werden kann.

Inhaltsverzeichnis

1 Einleitung 6

2 Theoretische Vorüberlegungen und Begrifflichkeiten 8
2.1 Vor- und Nachteile von Paarprogrammierung 8
2.2 Vor- und Nachteile von verteilter Paarprogrammierung 8
2.2.1 Awareness bei DPP 9
2.3 Sitzungstypen in PP- bzw. DPP-Sitzungen 11
2.3.1 Pair-Developing 12
2.3.2 Review 12
2.4 Wissenschaftliche Einbettung von Saros 14
2.5 Anwendung von Saros 16

3 Angewandte Forschungsmethoden 18
3.1 Canonical Action Research 18
3.2 Verfahren der Datenerhebung 20
3.2.1 Teilnehmende Beobachtung 20
3.2.2 Befragung 21
3.3 Verfahren der Datensicherung 26
3.3.1 Beobachtungsprotokoll 26
3.3.2 Technische Infrastruktur zur Beobachtung und Aufzeichnung 26
3.4 Verfahren der Datenauswertung 27

4 Beschreibung des Etablierungsprozesses 28
4.1 Projektbeteiligte Kooperationspartner 28
4.1.1 Das Unternehmen 28
4.1.2 Das Saros-Team 28
4.2 Planungsphase 29
4.3 Technische Infrastruktur der DPP-Sitzungen 30
4.4 Beteiligte Entwickler 31
4.4.1 Erfahrungen, Erwartungen und Motivation der Entwickler 31
4.4.2 Ziele der Entwickler 32
4.4.3 Rollen im Entwicklungsprozess 33
4.4.4 Aufgabenverteilung im Entwicklungsprozess 34
4.5 Sitzungstypen in den DPP-Sitzungen 35
4.5.1 Review 36
4.5.2 Pair-Developing 37
4.6 Ablauf und Organisation von Sitzungstagen und Maßnahmen 38
4.7 Überblick und Beschreibung der Phasen des Etablierungsprozesses 40
4.7.1 Pilotphase 42
4.7.2 Erster Arbeitszyklus 44
4.7.3 Zweiter Arbeitszyklus 47
4.7.4 Dritter Arbeitszyklus 50
4.7.5 Vierter Arbeitszyklus 52

5 Analyse des Etablierungsprozesses und des Nutzens von DPP 55
5.1 Abschließende Erhebung 55
5.2 Phänomene 57
5.2.1 Fehlende Abstimmung von Zielen 57
5.2.2 Ausgefallen 63
5.2.3 Verspätungen 74
5.2.4 Vorkommen und Fehlen von Awareness 84
5.3 Erkenntnisse des Etablierungsprozesses 90
5.3.1 Übersicht über aufgetretene Probleme 91
5.3.2 Erkenntnisse über den Nutzen von Fragebogen 92
5.3.3 Erkenntnisse über den Nutzen von Reflexionen 95
5.3.4 Weitere Erkenntnisse aus dem Etablierungsprozess 97
5.3.5 Erkenntnisse über erweiternde Maßnahmen im Etablierungsprozesses 98
5.3.6 Der Einfluss des Etablierungsprozesses auf die Entwicklung von Saros 99
5.4 Allgemeine Anleitung für einen optimierten DPP-Etablierungsprozess 101
5.5 Nutzen von DPP in der industriellen Softwareentwicklung 103

6 Resümee 107

7 Ausblick 108

8 Literaturverzeichnis 109

9 Glossar 113

Anhang 114
A. GQM-Planung 114
B. Fragebogen 116
C. Leitfäden 127
D. Saros-Funktionalitäten mit Screenshot 129
E. Anwendungsfälle 131
F. Detaillierte Beschreibung der Sitzungsverläufe 133

Abbildungsverzeichnis 137

Tabellenverzeichnis 138

1 Einleitung

In der Softwareentwicklung ist der Einsatz von agilen Softwareentwicklungsmethoden wie z.B. *Extreme Programming* (kurz XP) [Bec99] inzwischen keine Seltenheit mehr. Als Softwareentwicklungsmethode wird XP selbst von sehr großen Unternehmen, wie Microsoft eingesetzt, um den Erfolg ihrer aufwendigen Softwareprojekte zu sichern [Dwo09].

Paarprogrammierung (engl. *Pair Programming*, kurz PP) ist dabei eine etablierte Praktik, bei der zwei Entwickler gemeinsam Software entwickeln [Bec99]. Die Besonderheit dieser Methode ist, dass dafür nur ein Equipment (Rechner, Maus und Tastatur) verwendet wird. Die Entwickler wechseln sich bei der Bedienung des Equipments ab, woraus sich die Rollenverteilung ergibt. Der *Driver* ist derjenige, der das Equipment bedient, während seine Aktivitäten vom *Observer* verfolgt werden.
Verteilte Paarprogrammierung (engl. *Distributed Pair Programming*, kurz DPP) ist eine Variante von PP, bei der zwei Entwickler über eine räumliche Distanz gemeinsam Software entwickeln. Dies geschieht in derselben Rollenverteilung wie bei PP, jedoch hat jeder Entwickler ein eigenes Equipment. Stotts et al. [SWN+03] definieren es wie folgt:

"... two members of the team (whih may consist solely of these two people) synchronously collaborate on the same design or code from different locations. This means that both must view a copy of the same screen, and at least one of them should have the capability to change the contents on the screen."

Räumliche Distanz lässt sich heutzutage in global agierenden Softwareunternehmen nicht vermeiden. Die Entwickler eines gemeinsamen Projektes befinden sich bei der Zusammenarbeit an verschiedenen Orten, in verschiedenen Ländern oder sogar auf verschiedenen Kontinenten.
Ein Team, das sich durch die Anwendung von DPP zusammenfügt, kann als virtuelles Team (engl. *virtual Team*) bezeichnet werden. Bei einem virtuellen Team handelt es sich um eine Gruppe von Personen, die über verschiedene Zeitzonen verteilt und über kulturelle und organisatorische Grenzen hinweg auf ein gemeinsames Ziel hinarbeiten [GM02].
Unternehmen gründeten bereits vor mehr als 10 Jahren virtuelle Teams in der Erwartung, dass neue Technologien verteilte Softwareentwicklung ermöglichen würden [OO00]. Angesichts der Globalisierung wird es immer bedeutsamer Paarprogrammierung in verteilten Teams einzusetzen [BJ05].
Seit einigen Jahren beschäftigt sich auch die Wissenschaft mit dem Bedürfnis der Industrie nach globaler Softwareentwicklung [DL04]. Das Gebiet der agilen Softwareentwicklung ist jedoch bisher noch wenig erforscht. Die existierenden Studien wurden meistens in wissenschaftlichem Kontext und nicht in der Industrie durchgeführt und sind zum Teil bereits stark veraltet [MPT05]. Olsen und Olsen [OO00] nahmen bspw. vor etwa 10 Jahren wegen fehlender Technologien noch an, dass verteilte Softwareentwicklung nahezu unmöglich sei.

Besonders aus den genannten Gründen ist es interessant die Möglichkeit zu haben, DPP im industriellen Umfeld zu erforschen. Diese Arbeit beschreibt den Etablierungsprozess von DPP als Softwareentwicklungsmethode in einem internationalen Team von Entwicklern eines global agierenden Softwareunternehmens. Sie zeigt auf, dass die nötigen Technologien für DPP mittlerweile vorhanden sind und stellt dar, welche weiteren Erkenntnisse sich über DPP im industriellen Einsatz daraus ergeben.
Das Projekt entstand als Kooperation zwischen der Arbeitsgruppe *Software Engineering* der Informatik der Freien Universität Berlin (kurz AG SE) und der Teles AG (kurz Teles), einem mittelständischen Unternehmen in der Telekommunikationsbranche mit Niederlassungen in der ganzen Welt. Die AG SE stellte die nötigen Technologien zusammen und das DPP-Werkzeug Saros zur Verfügung und betreute den Etablierungsprozess. Das Unternehmen stellte seinerseits einige Entwickler aus Bangalore und Wien sowie deren Projekte zur Verfügung.

Die zentralen Fragen dieser Arbeit sind:

1. Welche Probleme können bei der Etablierung von DPP im industriellen Einsatz auftreten, und wie lassen sich diese durch Anpassung des Etablierungsprozesses beheben bzw. verringern?
2. Welchen Nutzen hat DPP im Rahmen industrieller Softwareentwicklung?

Aus den zentralen Fragen lassen sich folgende Aufgaben ableiten:
Zur Beantwortung der ersten Frage müssen die Probleme im Etablierungsprozess und Softwareentwicklungsprozess (weiterhin als Entwicklungsprozess bezeichnet) identifiziert und definiert werden und durch optimierende Eingriffe in ihrer Lösbarkeit geprüft werden. Die Beantwortung der zweiten Frage erfordert, die Erkenntnisse aus dem Etablierungs- und Entwicklungsprozess zu bewerten. Der Umfang des Nutzens resultiert aus der eindeutigen, zurechenbaren Verbesserung des Prozesses der Softwareentwicklung in virtuellen Teams durch die Anwendung von DPP.
Der Entwicklungsprozess bezieht sich auf Ereignisse innerhalb der DPP-Sitzungen, welche sich nur aus den Aktivitäten der Entwickler ergeben. Der Etablierungsprozess bezeichnet alle Aktivitäten und Ereignisse, die mit der Einführung von DPP als Softwareentwicklungsmethode zusammenhängen. Dies umfasst sowohl Ereignisse innerhalb der Sitzungen als auch alle weiteren Vorkommnisse, z.B. Interventionen, die sich durch den Einsatz von DPP ergeben.
Entwicklungs- und Etablierungsprozess beeinflussen sich und interagieren untereinander, daher sind sie beide Bestandteil des Forschungsprozesses.

Die angewandte Forschungsmethode basiert auf einem Ansatz nach *Canonical Action Research* (dt. kanonische Handlungs- oder Aktionsforschung, kurz CAR). Dies ist eine iterative Methode, die die allgemeine Herangehensweise strukturiert. Sie eignet sich für Forschungsprozesse wie in dieser Arbeit, bei denen eine Kombination von Beobachtung und Optimierung erforderlich ist. Es werden Erkenntnisse aus den gesammelten Daten verwendet, um konkrete Probleme des Entwicklungs- bzw. Etablierungsprozesses zu identifizieren und zu behandeln.
Zusätzlich wurde diese Methode um andere Forschungsmethoden zur Datenerhebung, -sicherung und -auswertung erweitert.

In Kapitel 2 werden die theoretischen Grundlagen dieser Arbeit beschrieben. In diesem Zusammenhang werden besondere Merkmale von PP und DPP sowie Vor- und Nachteile erläutert. Des Weiteren wird in diesem Kapitel das DPP-Werkzeug Saros wissenschaftlich eingebettet und die Funktionalitäten erläutert.

In Kapitel 3 werden die angewandten Forschungsmethoden vorgestellt. Insbesondere wird darauf eingegangen, inwiefern sie sich für den Forschungsprozess eignen und wie sie für die Planung und Ausführung der Forschungsaufgaben eingesetzt wurden.

In Kapitel 4 erfolgt die Beschreibung des Etablierungsprozesses. Ebenfalls werden in diesem Kapitel alle relevanten Informationen bzgl. des Projektes wie z.B. die Vorstellung der Beteiligten, die technische Infrastruktur und weitere Informationen wie z.B. über den Entwicklungsprozess aufgeführt.

In Kapitel 5 befinden sich Analysen über den Etablierungsprozess. Außerdem werden Erkenntnisse aufgeführt und daraus eine allgemeine Anleitung für einen Etablierungsprozess in DPP-Projekten abgeleitet. Weiterhin werden allgemeine Vor- und Nachteile von DPP im industriellen Umfeld beschrieben. Das Kapitel schließt mit einer Beschreibung des Einflusses des Etablierungsprozesses auf die Entwicklung von Saros.

In Kapitel 6 werden die gesammelten Erkenntnisse dieser Arbeit noch einmal zusammengefasst.

Kapitel 7 bietet einen Ausblick auf verschiedene Richtungen, in die sich zukünftige Projekte noch entwickeln könnten.

2 Theoretische Vorüberlegungen und Begrifflichkeiten

In diesem Kapitel werden Vor- und Nachteile von PP und DPP sowie besondere Merkmale davon vorgestellt. Danach werden die Sitzungstypen Pair-Developing und Review erläutert.
Das Kapitel endet mit einer wissenschaftlichen Einbettung und der Beschreibung der verwendeten DPP-Software Saros.

2.1 Vor- und Nachteile von Paarprogrammierung

PP ist die Grundlage für weitere Varianten von kollaborativer Softwareentwicklung wie z.B. DPP. Daher ist es relevant, kurz auf die wesentlichen Merkmale von PP einzugehen.

PP wird im Vergleich zu individueller Softwareentwicklung unter anderem nachgesagt, dass dadurch Software mit höherer Qualität erzeugt wird. Die gleichen Studien haben weiterhin gezeigt, dass es die Kommunikation und den Wissenstransfer im Team verstärkt, die Verantwortlichkeiten für die Artefakte dezentraler verteilt sind und Entwickler, z.B. durch *Pair-Pressure, Pair-Think, Pair-Review* und *Pair-Learning*, mehr Zufriedenheit bei der Entwicklung empfinden [CW01, Yap05, Wil00].
Pair-Pressure beschreibt den Effekt auf die Leistungsfähigkeit eines einzelnen, wenn er durch seinen Partner bei der Arbeit beobachtet wird. *Pair-Think* beschreibt die Abstimmung von verschiedenen Herangehensweisen bei der Erörterung von Entwurfsentscheidungen. *Pair-Review* beschreibt die kontinuierliche Überprüfung des Codes, die vorhanden ist, wenn zwei Entwickler gemeinsam Code erzeugen. *Pair-Learning* beschreibt den Lernprozess, den die Entwickler durch den gegenseitigen Erfahrungsaustausch beim gemeinsamen Entwickeln erleben.

Die meisten Studien wurden mit Studenten durchgeführt. Daher bezweifeln manche kritischen Betrachter, dass PP in der Praxis die gleichen Vorteile hat [MPT05]. Es gibt jedoch auch Studien mit ähnlichen Ergebnissen aus dem industriellen Bereich [Nos98]. Alle betrachteten Studien konnten bestätigen, dass die Qualität der erzeugten Software mit PP höher war. Einige Studien, die den Aufwand durch die doppelte Ressourcenbelegung bei PP untersuchten, kamen zu dem Ergebnis, dass PP im Vergleich zu individueller Softwareentwicklung einen Mehraufwand zwischen 15 % und 100 % erzeugen kann [Nos98, CW01, WKC00]. Weitere Vor- bzw. Nachteile von PP variieren z.B. abhängig von der Kooperations- und Kommunikationsbereitschaft der Entwickler, der Größe und Komplexität der Aufgabe sowie der Rollenverteilung im Team [WK00, MPT05]. Weitere Studien räumten ein, dass manche Vorteile sich erst nach einer Eingewöhnungsphase einstellten, über deren Dauer sie keine Aussage trafen [CW01, MPT05]. Diese Phase wird von Williams et al. [WKC00] als *Pair-Jelling* bezeichnet.

2.2 Vor- und Nachteile von verteilter Paarprogrammierung

Verschiedene Studien zeigten, dass DPP vergleichbare Vorteile hervorbringt wie PP und zusätzlich die Zusammenarbeit in virtuellen Teams dadurch gefördert wird [BWG+02, SWN+03, Yap05]. Nach Stotts et al. [SWN+03] kann DPP außerdem folgende Vorteile für virtuelle Teams haben: (1) Der Observer hat sein eigenes Equipment. Dadurch kann er die Aktivitäten an einem eigenen Bildschirm verfolgen und am eigenen Rechner unabhängige Aktivitäten, z.B. Recherche in einem Spezifikationsdokument, durchführen. (2) Man spart Kosten und Zeit der Entwickler, die sonst für eine Zusammenarbeit reisen müssten.
Extremfälle von virtueller Zusammenarbeit in globalen Unternehmen ermöglichen ein *around-the-clock XP-Team* [Yap05]. Dies ist ein Team, das verteilt auf drei verschiedene Kontinente rund um die Uhr Software entwickelt.

Wie die Definition von DPP von Stotts et al. [SWN+03] impliziert, ist es im Gegensatz zur räumlichen Nähe bei PP notwendig, dass die Zusammenarbeit bei DPP durch zusätzliche technische

Maßnahmen, z.B. eine Audioverbindung oder *Groupware*, unterstützt werden muss. Der Begriff der Groupware wird in Abschnitt 2.4 im Zusammenhang mit der wissenschaftlichen Einbettung von Saros erläutert. Es gibt verschiedene Ansätze zur Unterstützung von verteilter kollaborativer Softwareentwicklung. Manche der Ansätze basieren auf *Desktop Sharing* [MM02, RZ04]. Andere Ansätze wie Saros [DOS07] setzen auf eine integrierte Mehrbenutzer-Unterstützung.
Weiteren haben sich verschiedene Studien damit beschäftigt, welche die ideale technische Infrastruktur ist, z.B. ob die Notwendigkeit einer Videoübertragung besteht [SWN+03, Yap05, OO00]. Manche von ihnen behaupten bspw., der zusätzliche hohe technische Aufwand einer Videoübertragung würde sich nicht lohnen, weil kein Vorteil für die Paare daraus entstanden wäre [OO00, BWG+02]. Andererseits gibt es Studien, die belegen, dass Videoübertragung vorteilhaft für das Verständnis sei und das Feedback beschleunige [Yap05]. Andere Studien gehen davon aus, dass die passende technische Infrastruktur von den Präferenzen der jeweiligen Entwickler abhängt [SWN+03].
Meistens entstehen durch DPP auch zusätzliche Kosten, weil das nötige Equipment, z.B. Headsets, dafür noch angeschafft werden muss. Außerdem können bei technischen Ausfällen eines Rechners beide Beteiligten nicht weiterarbeiten.

Einer der größten Nachteile von DPP ist, dass nonverbale Kommunikation (ausgenommen Aktionen im Workspace, dt. Arbeitsplatz/Arbeitsbereich) meist nicht in die Ferne übertragbar ist [SWN+03]. Daher versuchen die meisten Ansätze, die Aktivitäten der anderen Benutzer durch sogenannte Awareness-Funktionalitäten so gut wie möglich sichtbar zu machen [SS01].
Wie Yap [Yap05] feststellt können durch DPP weitere Nachteile wie z.B. Entfallen von gemeinsamen Bürozeiten und gemeinsamen Pausen, die normalerweise die Teamentwicklung fördern, nicht kompensiert werden.
Zudem sind die Entwickler wegen der Distanz und dem Wegfall von nonverbaler Kommunikation darauf angewiesen, bei der Entwicklung ununterbrochen zu sprechen [SWN+03]. Auch kulturelle Unterschiede und die Verteilung auf verschiedene Zeitzonen können die Zusammenarbeit erschweren [Yap05].

2.2.1 Awareness bei DPP

Das Bewusstsein über Aktivitäten, die man bei PP durch räumliche Nähe erreicht, muss bei DPP über die Distanz ausgeglichen werden. Es ist aufgrund der Distanz im Gegensatz zu PP besonders wichtig zu wissen, wer anwesend ist, in welchem Teil des Arbeitsbereiches sich Anwesende befinden und woran sie gerade arbeiten. Das Verständnis in PP-Sitzungen bildet sich außer über Aktivitäten am Bildschirm noch durch Gestik, Mimik und andere Wahrnehmungen des Partners. Daher müssen auch diese so gut wie möglich in die Ferne vermittelt werden, um eine ähnlich effektive Kommunikationssituation zu schaffen [GG96].

Awareness (dt. Bewusstsein) beschreibt genau jenes Bewusstsein einer Person über Aktionen in ihrer Umgebung, wie es bei kollaborativer Softwareentwicklung nötig ist. Es wird von Dourish und Belotti [DB92] wie folgt definiert:

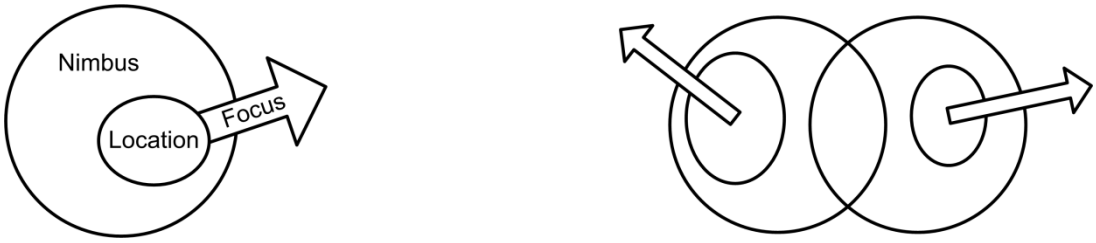
"Awareness is an understanding of the activities of others, which provides a context for your own activity"

Bei der Definition von Awareness nach Dourish und Belotti liegt der Schwerpunkt darauf, dass ein Verständnis für die Aktivitäten im Kontext einer Zusammenarbeit mit anderen Personen hergestellt wird. Kontext bezieht sich hierbei auf die Richtung, in die sich die Zusammenarbeit entwickelt und das jeweilige gemeinsame Ziel.
Dourish und Belotti benennen drei Vorteile von Awareness im Workspace: (1) es verringert Mehraufwand in der Zusammenarbeit, (2) es erzeugt neue Möglichkeiten der Zusammenarbeit und (3) es versorgt Personen mit mehr Informationen über den Kontext ihrer Aktionen.

Gutwin und Greenberg [GGR96] beziehen zusätzlich räumliche und zeitliche Aspekte in die Definition mit ein.

"workspace awareness, the up-to-the-moment understanding of another person's interaction with the shared space"

Das Modell von Rodden [Rod96] verdeutlicht zusätzlich zu den oben genannten Definitionen, welche Dimensionen Awareness bei einer räumlichen Betrachtung haben kann. Dieses Modell kann außerdem dazu verwendet werden, Awareness in verschiedene, quantifizierbare Stufen zu unterteilen. Nach Rodden besteht der Workspace aus gemeinsamen Objekten. Die Awarenessstufe zwischen zwei Personen bemisst sich nach der Anzahl sich überlappender Objekte im Workspace. Die Interessensgebiete einer Person unterteilt er dabei in: *Focus* (dt. Fokus), die Menge von Objekten, der man gerade seine Aufmerksamkeit schenkt, *Nimbus* (dt. Aura), die Menge von Objekten, mit der man möglicherweise noch interagieren könnte, und *Location* (dt. Ort), die Menge von Objekten, in der man selbst gerade präsent ist.



a) Location, Nimbus und Focus **b) Mode Overlap 1**

Abbildung 2.1 a) Location, Nimbus und Focus nach Rodden b) „Mode Overlap 1" des Awarenessmodells von Rodden

Abbildung 2.1a) zeigt die Beziehung zwischen den Interessensgebieten und Abbildung 2.1b) zeigt einen der 16 Modi, in die man die Interessensgebiete zweier kollaborierender Personen unterteilen kann. In diesem Modus sieht man *Location, Nimbus* und *Focus* von zwei Personen, deren Awareness sich nur im Bereich des Nimbus überschneidet. Man könnte dies bspw. so interpretieren, dass sie an verschiedenen Objekten arbeiten (Location), dabei ihre Aufmerksamkeit nicht auf die gleichen Objekte richten (Focus), aber Interaktionen zwischen den beiden in gleichen Objekten stattfinden könnten (Nimbus).

Abbildung 2.2 zeigt eine allgemeinere Betrachtung von Awareness, die neben räumlicher Awareness, wie Workspace Awareness, auch andere Sichtweisen auf kollaborative Zusammenarbeit berücksichtigt [GGR96]. Gutwin et. al. unterscheiden vier verschiedene Formen von Awareness, die sich überlappen und miteinander interagieren.

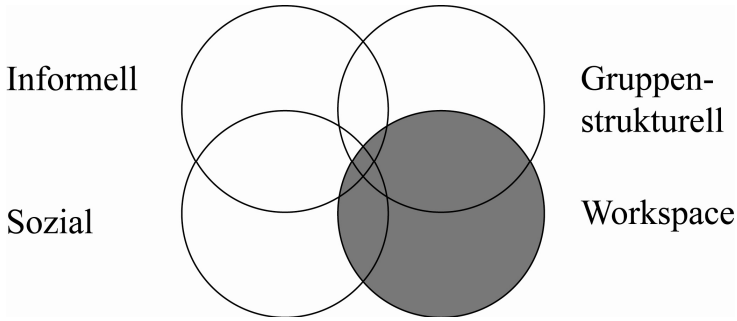


Abbildung 2.2 Arten von Awareness in kollaborativer Zusammenarbeit nach Gutwin et al. [GGR96]

Awareness über den Arbeitsbereich (engl. *Workspace Awareness*) wurde bereits in der zweiten Definition zu Awareness erwähnt. Es handelt sich dabei um Awareness bezogen auf Aktivitäten, die sich innerhalb des geteilten Workspaces ereignen, z.B. Informationen über Aktionen, Interaktionen und Absichten.
Informelle Awareness (engl. *Informal Awareness*) beschreibt allgemeine Informationen über Arbeitskollegen, z.B. deren Präsenz am Arbeitsplatz, ob sie erreichbar sind und mit welchen Aufgaben sie gerade beschäftigt sind.
Soziale Awareness (engl. *Social Awareness*) beschreibt alle Verhaltensweisen, die in einer Konversation mit anderen wahrgenommen werden können. Dazu gehören bspw. auch Emotionen, Interessen und Körpersprache.
Gruppenstrukturelle Awareness (engl. *Group-Structural Awareness*) beschreibt Awareness, die sich auf Rollen und Verantwortlichkeiten in der Zusammenarbeit beziehen.

Gutwin et al. [GGR96] formulieren Fragen entsprechend den Elementen von Awareness:

Element	Relevante Fragen
Identität	Wer nimmt an der Aktivität teil?
Standort	Wo sind sie?
Aktivitätsstufe	Sind sie im Workspace aktiv? Wie schnell arbeiten sie?
Aktivitäten	Was tun sie? Was sind ihre aktuellen Aktivitäten und Aufgaben?
Intentionen	Was werden sie tun? Wo werden sie sein?
Änderungen	Was für Änderungen nehmen sie vor? Wo werden Änderungen vorgenommen?
Objekte	Welche Objekte verwenden sie?
Ausmaße	Was können sie sehen?
Fähigkeiten	Was können sie tun?
Einflusssphäre	Worauf können sie Einfluss haben?
Erwartungen	Was brauchen sie von mir als nächstes?

Tabelle 2.1 Elemente von Workspace Awareness nach Gutwin et al. [GGR96]

Tabelle 2.1 bietet eine Übersicht über Elemente von Workspace Awareness. Es ist jedoch für verteilte kollaborative Zusammenarbeit nicht immer notwendig, dass alle Elemente unterstützt werden. In Abschnitt 5.2.4 wird näher darauf eingegangen, wie sich Awareness in den beobachteten Sitzungen gezeigt hat. In diesem Zusammenhang werden auch die Inhalte von Abbildung 2.2 und Tabelle 2.1 verwendet.

2.3 Sitzungstypen in PP- bzw. DPP-Sitzungen

Die Zeiträume, in denen sich Entwickler zusammensetzen, um PP bzw. DPP zu praktizieren, werden im weiteren Verlauf als DPP-Sitzungen bezeichnet. Ein Sitzungstyp beschreibt die inhaltliche Grundform einer DPP-Sitzung. Beispiele für Sitzungstypen sind Erzeugen von Quellcode (Implementierung), Durchsichten (Reviews) und Tests (Testen). Der Inhalt einer Sitzung bestimmt sich ganz allgemein nach dem Prozessmodell des Projektes und spezifisch nach den aktuellen Sitzungszielen.

Sitzungstypen haben nicht notwendigerweise eine feste Reihenfolge. Die Reihenfolge der Sitzungstypen hängt davon ab, wie der Softwareentwicklungsprozess gestaltet ist und welche Aktivitäten davon in Sitzungen durchgeführt werden. Manchmal gibt es innerhalb einer Sitzung auch verschiedene Sitzungstypen je nachdem welches Sitzungsziel verfolgt wird.
Betrachtet man eine Sitzung basierend auf einem sequentiellen Entwicklungsprozess, z.B. nach dem Wasserfallmodell [Boe76], könnte die Reihenfolge der Sitzungstypen in einer Sitzung so

aussehen: Erst wird das Artefakt erzeugt (Implementierung), danach wird es durchgesehen (Review) und am Ende wird getestet (Testen). Es wäre aber auch denkbar, dass zuerst getestet wird und später ein Review stattfindet.

Idealerweise sollten Entwickler alle Aktivitäten des Entwicklungsprozesses gemeinsam bearbeiten [WKC00]. Studien über den Einsatz in der Praxis haben jedoch gezeigt, dass dies nicht möglich und auch nicht immer nötig ist [Ave04].
In den folgenden Abschnitten werden nur solche Sitzungstypen näher erläutert, die für die Sitzungen des Etablierungsprozesses relevant waren.

2.3.1 Pair-Developing

Der Begriff *Pair-Developing* (dt. Entwicklung im Paar) ist sehr allgemein und beschreibt alle Aktivitäten, die dazugehören, wenn zwei Entwickler gemeinsam ein Artefakt, z.B. Quellcode, erzeugen [WK00]. Nach Williams und Kessler [WK00] sind darin die Analyse und der Entwurf (*Pair-Analysis* und *Pair-Design*), das Erzeugen eines Codes (*Pair-Implementation)* und gemeinsames Testen *(Pair-Testing)* enthalten.
Weitere Aktivitäten wie z.B. Ad-hoc-Tests, die die Entwickler zwischenzeitlich durchführen, um die Lauffähigkeit ihrer Programme zu testen, können darin ebenfalls enthalten sein.

Von Williams und Kessler [WK02] werden drei mögliche Erfahrungsstufen der Entwickler (*Expert, Average, Novice*) unterschieden, die in Pair-Developing Sitzungen kombiniert vorkommen können. *Expert* (dt. Experte) beschreibt die Rolle eines erfahrenen Entwicklers mit viel Domänenwissen und Entwicklungsgeschick. *Average* (dt. Durchschnitt) beschreibt einen Entwickler, der ebenfalls schon Erfahrungen mit Softwareentwicklung besitzt und bereits etabliert ist. *Novice* (dt. Novize) beschreibt entweder die Rolle eines Anfängers oder eines Entwicklers, dem die Erfahrung mit einer neuen Technologie noch fehlt.

Anhand der Kombinationen von Erfahrungsstufen, z.B. *Expert-Average* oder *Average-Average*, leiten Williams und Kessler [WK02] weitere Erkenntnisse über Pair-Developing ab. Betrachtet man bspw. die Kombination *Expert-Novice*, liegen die Vorteile dieser Rollenverteilung darin, dass der Novize durch die Teilnahme an der Sitzung etwas lernt, während der Experte trotzdem seiner Arbeit nachgehen kann. Der Novize profitiert davon, dass er sofort an der Entwicklung beteiligt ist. Auch der Experte profitiert davon. Er wird durch die Anwesenheit des Novizen gefordert, sein Vorgehen zu verlangsamen. Durch seine schrittweisen Erklärungen wird er dabei vom Novizen hinterfragt. Dadurch kann er seine Vorgehensweise prüfen und möglicherweise Fehlannahmen aufdecken.
Sowohl Williams und Kessler als auch andere Wissenschaftler beschäftigten sich ebenfalls mit den kritischen Aspekten der Kombinationen von Erfahrungsstufen [SR03, WK02]. Sie führen bspw. an, dass eine gute Zusammenarbeit von *Expert-Novice* voraussetzt, dass der Experte eine lehrende Rolle einnimmt und sich geduldig auf das Tempo des Novizen einstellt. Das Risiko ist jedoch sehr hoch, dass der Experte nicht genug auf den Novizen eingeht und ihn bei der Bearbeitung der Aufgabe außenvorlässt. Dies wir von Stephens und Rosenberg [SR03] als *Go Make Me a Cup of Tea-Syndrome* bezeichnet.

2.3.2 Review

Bei einer Durchsicht (engl. *Peer Review*, weiterhin auch als Review bezeichnet) untersucht mindestens eine andere Person (weiterhin als Reviewer bezeichnet) als der Autor selbst vorher erzeugte Artefakte, z.B. Quellcode. Dabei versucht der Reviewer bspw. Defekte aufzuspüren, indem er die Artefakte mit der Spezifikation abgleicht. Es ist wichtig, dass der Autor das Review nicht selbst durchführt, da der Reviewer eine neue Perspektive bzw. eine andere Denkweise in die Durchsicht mit einbringen soll. Diese Situation kennt jeder, der schon einmal versucht hat einen

selbstgeschriebenen Text zu korrigieren. Selbst sehr offensichtliche Fehler erkennt man erst, wenn ein anderer darauf hinweist.

Wiegers [Wie01] unterscheidet verschiedene Formen der Durchsicht anhand ihrer Formalisierungsgrade und ihrer Struktur. Abbildung 2.3 zeigt eine Übersicht über die verschiedenen Kategorien von Durchsichten anhand ihres Formalisierungsgrades.

Auf der linken Seite befinden sich die formalsten Varianten der Durchsicht, die Inspektion (engl. *Inspection*) und das Team-Review. Bei einer Inspektion gibt es einen Inspektionsplan und andere strukturierte Dokumente. Für die Durchsicht gibt es eigens definierte Ziele, feste Rollen in einem dafür geschulten Team und eine dokumentierte Vorgehensweise für den Ablauf. Team-Reviews sind etwas weniger formal. Sie sind jedoch aufwendig, denn sie setzen voraus, dass sich alle Stakeholder, z.B. auch Projektleiter, in einem Meeting zusammensetzen, um eine Durchsicht durchzuführen.
Auf der rechten Seite in Abbildung 2.3 stehen die am wenigsten formalen Formen der Durchsicht, das *Ad-hoc-Review* und der Schreibtischtest (*Peer Deskcheck* oder *Passaround*). Ein Ad-hoc-Review kommt immer dann zustande, wenn der Autor eines Artefaktes spontan einen Kollegen über sein Artefakt schauen lässt, weil er ein unmittelbares Problem hat. Hierfür gibt es keine Formalien wie bei der Inspektion: weder wird etwas explizit geplant, noch gibt es neben dem Artefakt andere Dokumente oder feste Rollen. Ein Schreibtischtest ist ebenfalls wenig formal. Der Reviewer prüft die Korrektheit des Programms unabhängig vom Autor und meistens ohne die Hilfe eines Rechners.

Umso weiter man sich auf der Skala von Abbildung 2.3 in die Mitte bewegt, desto näher kommt man dem Bereich von Aktivitäten, die auch in DPP-Sitzungen vorkommen können. Da gibt es zum einen den *Walkthrough* und zum anderen *Pair Programming* (bzw. das dort enthaltene *Pair-Review*), die beide nicht so formal wie eine Inspektion, aber auch nicht so spontan wie ein Ad-hoc-Review sind.
Ein Walkthrough ist eine wenig formale Durchsicht, bei der ein Autor einer Gruppe von anderen Personen sein Artefakt zeigt, damit diese ihm dazu ihre Kommentare geben können. Im Unterschied zur Inspektion sollen damit hauptsächlich Ziele des Autors verfolgt werden und der Ablauf ist ohne formale Vorgaben. Für einen Walkthrough ist zwar Planung vorgesehen, aber es ist keine bestimmte Art der Vorbereitung wie z.B. der Entwurf von Review-Dokumenten, nötig. Es gibt nach der Korrektur der gefundenen Verbesserungsmöglichkeiten auch keine Verifikationsphase, wie es bei der Inspektion üblich wäre.
Bei PP ergibt sich das Pair-Review zwangsläufig während der PP-Sitzung. Da zwei Entwickler beteiligt sind, gibt es neben dem Autor (Driver) auch immer einen Reviewer (Observer), sodass immer zwei Entwickler gleichzeitig über das erzeugte Artefakt schauen. Auf diese Weise wird kontinuierlich schon während des Erzeugungsprozesses eine Durchsicht durchgeführt. Diese Form der beiläufigen Durchsicht ist unstrukturiert. Es werden dafür nicht noch zusätzliche Abläufe geplant, Dokumente erstellt oder andere Vorbereitungen getroffen. Der Nachteil an dieser Form der Durchsicht ist, dass die Außenansicht fehlt, d.h. der Abstand zwischen demjenigen, der gerade die Rolle des Reviewers einnehmen soll, zum erzeugten Artefakt ist zu gering.

Der Unterschied zwischen Walkthroughs und Pair-Reviews ist, dass es sich bei Pair-Reviews nicht um einen eigenen Sitzungstyp handelt. Pair-Reviews treten als Nebeneffekt bei der Softwareentwicklung in Sitzungen vom Typ Pair-Developing auf, während Walkthroughs eine eigenständige Sitzungsaufgabe beschreiben, bei der die Korrektheit eines bereits existierenden Artefaktes geprüft wird.

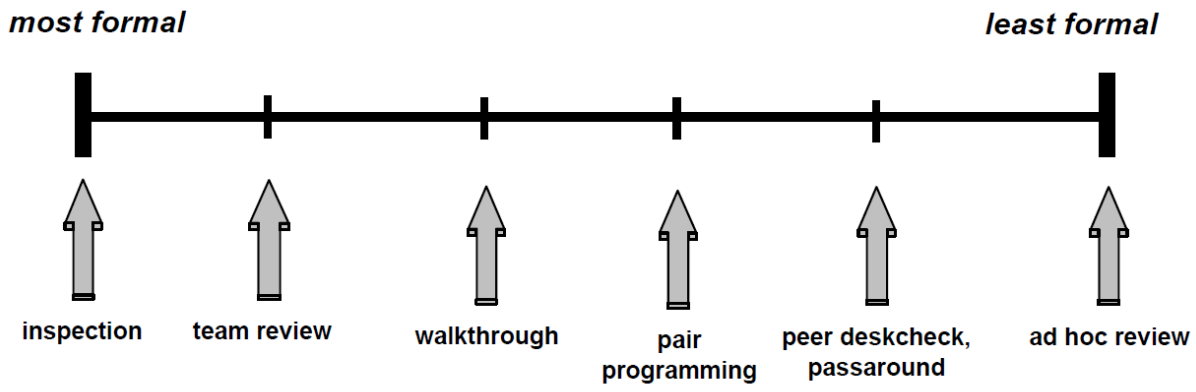


Abbildung 2.3 Formalitätsgrade von Durchsichten nach Wiegers [Wie01]

2.4 Wissenschaftliche Einbettung von Saros

Saros ist eine Software, die zur verteilten kollaborativen Softwareentwicklung eingesetzt werden kann. Der Prototyp wurde 2006 im Rahmen einer Diplomarbeit, betreut durch die Arbeitsgruppe Software Engineering (kurz, AG SE), entwickelt [Dje06]. Die Software wird in ein anderes Programm integriert, um internetbasierte kollaborative Zusammenarbeit in Echtzeit zu unterstützen. Die Informationen zwischen den Beteiligten werden über Texteditoren geteilt.

In diesem Abschnitt wird erläutert, in welche Konzepte Saros wissenschaftlich eingebettet ist und welche anderen Arbeiten auf diesem Gebiet vorliegen.
Saros ist ein Werkzeug, das sich in eine Entwicklungsumgebung einbetten lässt, um verteilte Paarprogrammierung mit Hilfe von kollaborativen Texteditoren zu unterstützen. Es handelt sich dabei um eine Groupware. Ellis et al. [EGR91] haben dies folgendermaßen definiert:

"computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment."

Bei dieser Aussage liegt der Schwerpunkt auf zwei Aspekten. Erstens geht es darum, dass mehrere Personen eine gemeinsame Aufgabe erledigen. Zweitens geht es darum, dass sie dies in einer geteilten Umgebung tun. Um zwischen verschiedenen Arten von Groupware, z.B. Chat, Emails oder Wikis,
unterscheiden zu können, haben Ellis et al. [EGR91] sie nach zeitlichen und örtlichen Kriterien ihrer Anwendungsweise gruppiert. Demnach würde Saros im Gegensatz zu anderer Groupware als *synchronous distributed interaction* [EGR91] bezeichnet werden, da die Bearbeitung zeitlich synchron und an verschiedenen Orten geschieht.

Die Forschung und Entwicklung zu Groupware gehört in den Bereich der *Computer Supported Cooperative Work* (kurz CSCW). CSWS unterscheidet die Ansätze *Collaboration Awareness* und *Desktop Sharing* für verteilte kollaborative Zusammenarbeit.
Desktop Sharing (oder *Screen Sharing*) Anwendungen unterstützen die Zusammenarbeit, indem sie die Ansicht des Desktops einer Person auf die Systeme anderer Personen übertragen. Dabei bleibt die Kontrolle über Aktivitäten im Workspace immer bei einer der beteiligten Personen. Desktop Sharing Anwendungen sind z.B. Microsoft Netmeeting [Netmeeting] oder Netviewer Meet [Netviewer]. Eine frei verfügbare Variante ist z.B. Virtual Network Computing (kurz VNC) [UltraVNC].
Im Gegensatz dazu wird bei Anwendungen der Kategorie *Collaboration Awareness* den Benutzern die Kontrolle nicht entzogen oder erteilt, sondern es ist eine Mehrbenutzer-Unterstützung in der Software integriert. Neben Saros gibt es in diesem Bereich mehrere Softwareprojekte, die

einen ähnlichen Ansatz verfolgen. Dazu gehört z.B. Google Docs [GoogleDocs] und auch andere Opensource-Projekte wie z.B. PEP, Sangam, XecliP und XPairtise [PEP, Sangam, XecliP, XPairtise].
Für die Verwendung von Texteditoren zur verteilten kollaborativen Softwareentwicklung gibt es neben Desktop Sharing und Entwicklungsumgebungen (engl. *Integrated Development Environment*, kurz IDE) mit Mehrbenutzer-Unterstützung noch kollaborative Texteditoren als dritte Alternative. Ein Beispiel für verteilte kollaborative Texteditoren ist die Arbeit im Projekt SubEthaEdit. Damit können Benutzer über einen Server gemeinsam an Textdokumenten arbeiten [SubEthaEdit]. Diese Anwendung ist jedoch nicht frei verfügbar und nicht plattformunabhängig, sondern kann bisher nur von Anwendern mit MAC-Betriebssystemen genutzt werden.

Die Wahl der Groupware (ob Desktop Sharing oder Collaboration Awareness) kann von der geplanten Aufgabe abhängen.
Durch Collaboration Awareness-Werkzeuge werden nur Aktionen innerhalb des Workspace geteilt, d.h. Benutzer haben weiterhin die Möglichkeit, sich außerhalb des Projektes frei auf ihrem System zu bewegen. Diese Alternative ist effizienter in der Datenübertragung und gewährt mehr Privatsphäre, da nur die für die Zusammenarbeit nötigen Daten übertragen werden. Doch ist Voraussetzung für den Einsatz, dass die Benutzer dafür die gleiche Software installiert haben. Weiterhin können aufgrund der Distanz und der eingeschränkten Einsicht in die Handlungen der anderen Benutzer Probleme mit Awareness aufkommen. Bei Awareness handelt es sich um das Bewusstsein über die Aktivitäten anderer Benutzer im Workspace. Diese Problematik wurde bereits näher in Abschnitt 2.2.1 erläutert.
Es kann daher gewünscht sein, Desktop Sharing zu verwenden, um zu garantieren, dass die Benutzer dieselbe Ansicht haben. Dies kann jedoch zu doppelten Daten und Konsistenzproblemen führen. Es werden zudem eventuell mehr Daten als nötig zwischen den Beteiligten geteilt, weil auch Fenster übertragen werden, die nichts mit der aktuellen Arbeit zu tun haben. Dadurch ist erstens mehr Aufwand für die Übertragung der Daten nötig und zweitens wird evtl. weiter in die Privatsphäre der Beteiligten eingegriffen als nötig.
Für verteilte kollaborative Softwareentwicklung eignen sich daher Anwendungen, die Mehrbenutzer-Unterstützung haben, besser [SWN+03].

Bei Saros handelt es sich um ein Plug-In für *Eclipse*. Es ist ein Collaboration Awareness-Werkzeug. Eclipse ist eine freiverfügbare IDE, die von Entwicklern zur Softwareentwicklung eingesetzt werden kann [Eclipse]. Saros ermöglicht es Eclipse-Anwendern, Informationen über die Anzeige von gleichen Editorfenstern miteinander zu teilen.
Saros wurde unter der Lizenz GNU – General Public License (GPL) als *Open Source Sofware* (kurz OSS) entwickelt. Dies bedeutet, dass Autoren alle Artefakte ihrer Software öffentlich zur freien Verfügung stellen. Insbesondere ist der Quellcode für alle Benutzer frei verfügbar. Open Source Projekte zeichnen sich unter anderem dadurch aus, dass die Benutzer hohen Einfluss auf die Entwicklung der Funktionalitäten haben. Benutzer können nach einer Prüfung durch bestimmte Projektmitglieder auch eigene Änderungen in die vorhandene Software einbringen [OSI].

Saros unterstützt durch die vorhandenen Funktionalitäten wie z.B. dem parallelen Mehrbenutzerbetrieb (engl. *Multi-Driver-Mode*) oder der Möglichkeit mehr als einen Observer zu beteiligen (siehe 2.5) mehr Aktivitäten, als die Definition von PP und DPP enthalten. Nach diesem Verständnis würde sich ein allgemeinerer Oberbegriff zur Beschreibung der Methodik wie z.B. verteilte kollaborative Softwareentwicklung besser eignen. Da darin auch weitere Varianten von kollaborativer Softwareentwicklung wie z.B. *Side-by-Side Programming* (kurz SbS) enthalten wären. Bei SbS teilen sich die Entwickler ihre Aufgabe auf und bearbeiten die Teilaufgaben mit getrenntem Equipment unabhängig voneinander, bleiben dabei aber stets in enger Kooperation [Coc04].
Da die Sitzungen im Etablierungsprozess, trotz der Möglichkeiten von Saros, jedoch hauptsächlich mit zwei Entwicklern stattfanden und diese meistens gemeinsam arbeiteten, wird im weiteren Verlauf dieser Arbeit die Bezeichnung DPP verwendet werden.

2.5 Anwendung von Saros

Saros funktioniert unabhängig von Betriebssystemen und unterstützt verschiedene Programmiersprachen, z.B. Java, C++ via CDT, Python via PyDev und PHP via PDT. In jedem der im Abstand von drei Wochen stattfindenden Releases werden Änderungen und Erweiterungen, z.B. auch für die Unterstützung von weiteren Programmiersprachen, veröffentlicht.
Bei der Anwendung von Saros gilt der Replikationsansatz, d.h. vor der Verwendung in einer DPP-Sitzung haben alle Beteiligten die gleichen Informationen vorliegen und können den Quellcode in ihrer eigenen Entwicklungsumgebung übersetzen lassen und auch ausführen [Dje06]. Während der Verwendung von Saros zeigt der *Watchdog* (der Konsistenzprüfer) an, falls die angezeigten Informationen inkonsistent sind, d.h. sich zwischen den Benutzern unterscheiden. Außerdem wird automatisch dafür gesorgt, dass die Inkonsistenzen aufgelöst werden.
Im Gegensatz zu vielen anderen Programmen seiner Art unterstützt Saros parallelen Mehrbenutzerbetrieb (engl. *Multi-Driver Mode*). Damit ist es möglich, dass mehrere Benutzer gleichzeitig eine Datei editieren. Außerdem ist die Anzahl der Benutzer, die an einer Sitzung teilnehmen, nicht begrenzt.
Eine Übersicht über die wichtigsten Funktionalitäten, ein Screenshot der IDE inklusive Saros sowie Anwendungsfälle zu Sitzungstypen (Review und Pair-Developing) befinden sich in Anhang D und E. Der weitere Abschnitt beschreibt detailliert die Funktionsweise von Saros.

Nach der Installation richtet man sich zuerst einen Benutzernamen auf einem beliebigen Kommunikationsserver (Jabber/XMPP-Server) ein, den man verwenden will, um mit seinen *Buddies* (andere Anwender von Saros, mit denen man Editorfenster teilen möchte) zu kommunizieren. Man kann sich entweder vorher einen eigenen Server einrichten, oder man verwendet einen öffentlichen Server, z.B. jabber.cc.
Saros hat zwei Views, den *Roster* und *Shared Project Session*. Views sind Fenster, die nach Bedarf in der IDE angezeigt oder ausgeblendet werden können.
Hat man den Benutzer erfolgreich eingerichtet, erscheint der Roster. Der Roster ist ein Fenster innerhalb von Eclipse, in dem Informationen über den Kommunikationsstatus und die Buddies erscheinen, z.B. welche Buddies man schon einmal kontaktiert hat, ob man gerade mit dem Server verbunden ist und welche Buddies auch verbunden sind. Im Roster wird auch angezeigt, welche Verbindungsqualität für die Verbindung mit anderen Buddies im Netz besteht.
Für das Teilen von Editorfenstern mit Saros ist es erforderlich, während der gesamten Zeit, in der man gemeinsam Editorfenster betrachten möchte, mit dem Server und den Buddies verbunden zu sein.
Der nächste Schritt ist also, sich über den Verbindungsknopf im Roster mit dem Kommunikationsserver (z.B. Jabber.cc) zu verbinden.
Sobald man mit dem Server verbunden ist, erscheint der Buddiesknopf, und man kann anfangen, mit seinen Buddies Kontakt aufzunehmen. Die Kontaktaufnahme geschieht durch einen eigenen Einladungsdialog. Es wird dann eine Nachricht an den gewünschten Buddy verschickt. Voraussetzung hierfür ist, dass man dessen Benutzer und die Serveradresse dazu kennt (z.B. username@jabber.cc).

Die oben beschriebenen Einstellungen müssen nur einmal beim Einrichten von Saros durchgeführt werden. Um danach Saros zu verwenden, reicht es, sich mit dem Verbindungsknopf (im Roster) auf dem Server zu verbinden und die Buddies auszuwählen, mit denen man seine Informationen teilen möchte.
Die Informationen, welche bei einem Benutzer auf verschiedene Editordateien verteilt sind, werden von Eclipse als Projekte verwaltet. Zum Austausch von Informationen verschickt Saros eines dieser Projekte zwischen den Benutzern am Anfang jeder Saros-Sitzung. Der Benutzer, der sein Projekt mit den anderen teilt, wird *Host* genannt.
Es gibt in Saros entsprechend des Konzeptes von PP bzw. DPP zwei verschiedene Rollen. Der Driver, der durch die Dateien navigiert und Änderungen am Projekt vornehmen kann, und der Observer, der die Aktivitäten des Drivers verfolgt.

Nach Eröffnung einer Saros-Sitzung befinden sich alle weiteren Funktionalitäten von Saros in der zweiten Ansicht „Shared Project Session“. Es können jederzeit weitere Personen zur Sitzung eingeladen werden, oder Benutzer können die aktuelle Sitzung wieder verlassen. Dafür dienen die „Zur Sitzung einladen“ und „Sitzung verlassen“ Knöpfe. Während einer Sitzung können beliebig oft die Rollen von Driver und Obsever gewechselt werden. Durch die Vergabe der exklusiven Driver-Rolle bleibt weiterhin nur ein Driver in der Sitzung. Es gibt aber auch die Möglichkeit, die Driver-Rolle an mehrere Benutzer zu vergeben (Multi-Driver Mode).
In einer Saros-Sitzung können alle üblichen Editorfunktionalitäten genutzt werden. Benutzer können bspw. Dateien öffnen und schließen, Änderungen vornehmen und diese dann abspeichern oder verwerfen. Es ist jederzeit möglich alle Vorteile der IDE wie z.B. automatische Codegenerierung zu verwenden oder Testläufe zu starten.
In Saros wurden verschiedene Mechanismen eingebaut, die die Awareness zwischen den Beteiligten unterstützen sollen (siehe Screenshot im Anhang D). Dazu gehören verschiedene farbliche Markierungen innerhalb und um die Editorfenster herum, die anzeigen, wo sich andere Benutzer gerade befinden. Eine weitere wichtige Funktionalität für die Unterstützung der Awareness in verteilter Zusammenarbeit ist der Verfolgermodus. Dieser kann in der Shared Project Session-Ansicht nach Bedarf ein- oder ausgestellt werden. Er sorgt dafür, dass alle Aktivitäten des Drivers für den Observer sichtbar sind, indem automatisch alle Dateien geöffnet werden, die beim aktuellen Driver offen sind, und die Stelle angezeigt wird, an der dieser sich gerade befindet.

Alle weiteren Informationen zum Projekt und der Software sind über die offizielle Internetseite[1] einsehbar.

[1] https://www.inf.fu-berlin.de/w/SE/DPP

3 Angewandte Forschungsmethoden

Für den Forschungsprozess standen alle notwendigen Ressourcen zu Beginn des DPP-Projektes bereit. Es gab anfangs keine weiteren Vorgaben für das Projekt, als die Etablierung von DPP und eine bestmögliche Betreuung dafür. Wie genau der Etablierungsprozess gestaltet sein würde, war zu diesem Zeitpunkt noch unklar. Sowohl die Forschungsmethode als auch die Sitzungsinhalte waren experimentell.
Diese Arbeit basiert auf einem qualitativen empirischen Forschungsansatz. Es handelt sich dabei um ein Feldexperiment, da die Entwickler in ihrer realen Arbeitsumgebung beobachtet werden. Es hat auch den Charakter einer explorativen Einzelfallstudie, weil im Vorhinein die Entwicklung des Projektes nicht abzusehen war. Der Etablierungsprozess geht mit einer Innovationseinführung wie von Denning und Dunham [DD06] beschrieben einher, denn die Softwareentwickler hatten vorher weder die für DPP erzeugte Software noch DPP zur Softwareentwicklung angewendet.
Die angewandte Forschungsmethode soll die Bearbeitung der in der Einleitung beschriebenen Aufgaben bestmöglich unterstützen. Daher wurde für das grundlegende Vorgehen und die Struktur des Forschungsprozesses CAR ausgewählt. Die iterative Methode sorgt dafür, dass im Etablierungsprozess abwechselnd Datenerhebung und -auswertung stattfindet, sowie ebenfalls die Möglichkeit für die Ergreifung von Maßnahmen besteht.
Zusätzlich zu CAR wurden weitere Methoden zur Datenerhebung (GQM, etc.) und zur abschließenden Datenauswertung (qualitative Inhaltsanalyse) angewendet.

In den folgenden Abschnitten werden der allgemeine Forschungsprozess von CAR und alle weiteren für diese Arbeit relevanten Forschungsmethoden erläutert.

3.1 Canonical Action Research

Canonical Action Research basiert auf der 1947 von Kurt Lewin entwickelten Methode der Handlungsforschung.

„In der Handlungsforschung sind jene Menschen und Menschengruppen, welche von den Wissenschaftlern untersucht werden, nicht mehr bloße Informationsquelle des Forschers, sondern Individuen, mit denen sich der Forscher gemeinsam auf den Weg der Erkenntnis zu machen versucht.“ [Mos77]

Die Beschreibung von Moser enthält bereits die wesentlichen Aspekte von Handlungsforschung. Der Abstand zwischen Forscher und Forschungsfeld wird verringert und damit werden die Teilnehmer des Forschungsprojektes zu Beteiligten am Forschungsprozess.

Huschke-Rhein [Hus87] formulierte folgende Grundsätze der Handlungsforschung:

Forschungskontext:
- Das Forschungsfeld bezieht sich auf Geschehnisse im sozialen oder erzieherischen Feld.
- Die Forscher sind nicht distanzierte Beobachter, sondern partizipieren durch teilnehmende Beobachtung am Forschungsprozess.
- Die Teilnehmer und ihre Situation bilden das Forschungsobjekt.
- Die gewählten Interaktionsformen bilden die Basis für den Verlauf und die Veränderungen im Forschungsprozess.

Forschungsverlauf:
- Die Methoden werden anhand ihrer Notwendigkeit für den Forschungsprozess gewählt.
- Die Planung wird auch an den Wünschen und Bedürfnissen der Teilnehmer ausgerichtet.
- Der Projektablauf hat einen zyklischen Verlauf, und währenddessen werden Teilergebnisse den Teilnehmern kommuniziert, die zu Änderungen im Projektplan führen können.

Forschungsergebnisse:

- Die Auswertung bezieht sich auf die Analyse des gesamten Forschungsprozesses und nicht auf die Verifikation von zuvor aufgestellten Hypothesen.
- Aufgestellte Hypothesen dienen als Grundlage für weiterführende Erkenntnisse über die Situation.
- Die Gütekriterien für den Forschungsprozess sind Realitätsnähe, Transparenz, Praxisrelevanz und Interaktion.

Ein besonders wichtiger Aspekt dieser Grundsätze betrifft die Gütekriterien. Durch die Anwendung von Handlungsforschung wird nicht primär versucht, allgemeingültige Erkenntnisse aufzustellen. Vielmehr werden komplexe soziale Interaktionen aus der Praxis untersucht und auch verändert.

CAR nach Davison et al. [DMK04] versucht im Gegensatz zum allgemeinen Ansatz der Handlungsforschung, durch fünf Prinzipien das Vorgehen des Forschungsprozesses zu strukturieren. Es handelt sich um einen iterativen Prozess, der in einzelne Prozessschritte unterteilt ist (siehe Abbildung 3.1). Die Iterationen bauen aufeinander auf. Durch jede Iteration werden neue Erkenntnisse gesammelt, die in die nächste Iteration einfließen. Innerhalb der Iterationen werden Erkenntnisse über den Prozess gesammelt und Maßnahmen zur Optimierung geplant. Durch das Ergreifen von Maßnahmen findet eine Rückkopplung des Informationsflusses in den Prozess statt. Sowohl die Anzahl der Prozessschritte, als auch die Länge der Iterationen variieren je nach Bedarf des Forschungsprozesses.
Erstes Prinzip ist das *Researcher–Client Agreement* (dt. Forscher-Teilnehmer-Abkommen, kurz RCA). Im RCA werden die Bedingungen für die gemeinsame Arbeit von Forscher und Teilnehmern definiert. Dabei werden gemeinsame Ziele festgelegt, Rollen und Verantwortlichkeiten geklärt sowie die Bedingungen für den Etablierungsprozess besprochen.
Zweites Prinzip ist das *Cyclical Process Model* (dt. zyklisches Prozessmodell, kurz CPM). In Abbildung 3.1 wird der empfohlene zyklische Prozess gezeigt, der CAR zugrunde liegt. Es gibt darin fünf Stufen (Diagnose, Aktionsplanung, Ergreifen von Maßnahmen, Auswertung und Reflexion), die sequentiell und unidirektional durchgeführt werden sollen. In einem Forschungsprojekt gibt es normalerweise mehr als eine Iteration. Davison et al. schließen dabei nicht aus, dass es innerhalb des Zyklus zu Variationen der Reihenfolge und der Stufen kommen kann.

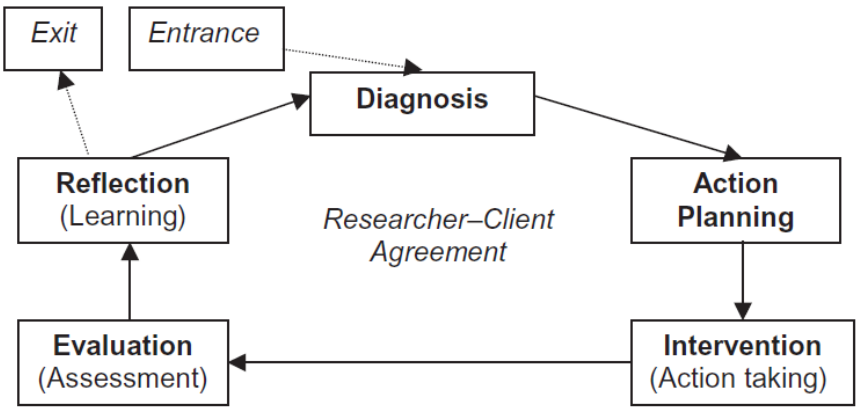


Abbildung 3.1 CAR-Prozessmodell nach Davison et al. [DMK04]

Das dritte Prinzip, *Theory* (dt. Theorie), beschreibt den wissenschaftlichen Hintergrund, auf dem die Forschung basieren sollte. Davison et al. räumen ein, dass es nicht immer möglich ist, vor Beginn des Projektes eine definierte Theorie zu entwickeln. Es ist jedoch notwendig, über ausreichend Hintergrundwissen auf dem Forschungsgebiet zu verfügen und die Relevanz der Forschung für die Wissenschaft und die Teilnehmer sicherzustellen.
Das vierte Prinzip, *Change through Action* (dt. Veränderung durch Aktion), beschreibt eines der Kernelemente von CAR. Gerade durch dieses Prinzip unterscheidet sich CAR von anderen For-

schungsmethoden. Maßnahmen des Forschers sollen dafür sorgen, dass Situationen von unerwünschten Eigenschaften befreit werden. Dieses Prinzip unterstützt folgende Ideen: (1) Die Motivation für Änderungen muss sowohl beim Forscher als auch bei den Teilnehmern vorhanden sein. (2) Die vom Forscher gewählten Interventionen können experimenteller Natur sein und sollten je nach Bedarf der Situation flexibel eingesetzt werden.
Im fünften und letzten Prinzip, *Learning through Reflection* (dt. Lernen durch Reflexion), soll die Verantwortung des Forschers im Umgang mit Ergebnissen der Forschung und Zielen der Teilnehmer geregelt werden. Demnach sollen Erfahrungen und erlangtes Wissen aus dem Forschungsprozess in regelmäßigen Abständen gegenüber den Teilnehmern kommuniziert werden und auch in weitere Aktionen einfließen.

3.2 Verfahren der Datenerhebung

Daten wurden auf drei Arten erhoben: (1) als teilnehmende Beobachtung in den Sitzungen, (2) als Fragebogen und (3) als Einzel- oder Gruppengespräche.

3.2.1 Teilnehmende Beobachtung

Eines der wichtigsten Mittel zur Ergründung in der Handlungsforschung ist die teilnehmende Beobachtung wie sie von Huschke-Rhein beschrieben wird [Hus87]. Neben den Beobachtungen wird durch Maßnahmen im Rahmen der Betreuung aktiv in den Forschungsprozess eingegriffen. Dies ermöglicht eine Symmetrie in der Kommunikation zwischen allen Beteiligten.
Eines der Merkmale der durchgeführten Beobachtung ist die niedrigstrukturierte Vorgehensweise. Der Vorteil davon ist, dass dadurch genug Offenheit vorhanden ist, um neue Problematiken entdecken zu können.
Während des Etablierungsprozesses vereinfachte eine Einteilung in drei Kategorien (organisatorische, inhaltliche und technische) die Beobachtung. Organisatorische Aspekte bezogen sich auf Maßnahmen des Etablierungsprozesses. Inhaltliche Aspekte bezogen sich auf Geschehnisse zwischen den Entwicklern in den Sitzungen und technische Aspekte bezogen sich auf alle Vorkommnisse, die mit der technischen Infrastruktur zu tun hatten.
Aufgrund der Konstellation der Beobachtung (internetbasierte Sitzungen mit Audiokonferenz) basierte die Beobachtung besonders auf Fernsinnen. Es war möglich zu hören, was die Entwickler sagten, und eingeschränkt auch deren Aktivitäten am Rechner zu sehen. Nahsinne wie z.B. Fühlen oder Riechen bzw. die Entwickler persönlich zu beobachten konnten nicht genutzt werden.
Die Teilnahme an Sitzungen war stets mit dem Fokus auf den Etablierungsprozess gerichtet. Es gehörte bspw. zu meinen Aufgaben, Funktionalitäten der verwendeten technischen Infrastruktur (siehe Kapitel 4.3), die für die Abwicklung der Sitzungen eingesetzt wurde, zu erläutern oder bei technischen Problemen ein alternatives Vorgehen anzubieten. Im Gegensatz dazu gehörte es nicht zu meinen Aufgaben, den Entwicklern bei konkreten Entwicklungsproblemen wie z.B. Fehlermeldungen, Defekten oder Schreibfehlern zu helfen.
Des Weiteren versuchte ich mich weitestgehend aus der Planung der Sitzungsinhalte herauszuhalten.
Weniger strukturierte Beobachtungen können im Allgemeinen verschiedene Nachteile haben. Sie sind in der Durchführung eventuell weniger exakt. Auch die Ergebnisse dieser Beobachtungsform sind womöglich weniger vergleichbar, weil sich ständig der Fokus ändern kann. Die Beobachtungen sind außerdem schwieriger, vor allem weil sie besondere Anforderungen an die Qualifikation des Beobachters voraussetzen.
Es bestand die Gefahr, dass diese Form der teilnehmenden Beobachtung das Verhalten der Entwickler möglicherweise beeinflussen könnte. Demgegenüber bot sich so die Gelegenheit, bei den DPP-Sitzungen unmittelbar dabei zu sein.
Die aktive Teilnahme des Beobachters kann in manchen Fällen die Objektivität der Beobachtung und die nötige Distanz zum Beobachtungsobjekt stören. In diesem Fall bestand wenig Gefahr durch zu wenig Distanz, denn in den Sitzungen wurden spezielle Anwendungen entwickelt, die

spezifische Vorkenntnisse voraussetzten, über die ich nicht verfügte. So konnte ich mich auf die Beobachtung des Prozesses konzentrieren.

3.2.2 Befragung

Die Forschung nur über die Beobachtung zu gestalten (von Huschke-Rhein [Hus87] als *Behaviorismus* bezeichnet), reichte nicht aus, um alle relevanten Informationen über den Etablierungsprozess einzuholen. Um die Situationen richtig einschätzen zu können, waren auch Informationen über Motive, Einstellungen und Überzeugung der Entwickler nötig. Daher wurde die Datenerhebung um zusätzliche Befragungen zu jeder Sitzung erweitert. Die Befragungen fanden in schriftlicher Form durch Fragebogen und mündlich in Form von Gruppendiskussionen als Teil der Reflexionen oder als Einzelgespräche statt. In diesem Abschnitt wird nur über die Zielsetzung und die Planung der Befragung gesprochen. Abläufe und Ergebnisse werden in den Kapiteln 4 und 5 erörtert.

3.2.2.1 Zielsetzung und Planung der schriftlichen Befragung

Die Fragebogen sollten zusätzliche Informationen über zwei Aspekte erbringen: Einerseits wurde dadurch eine Übersicht über Schwerpunkte und Ziele der Sitzung aus Sicht der beteiligten Entwickler erwartet, andererseits ging es darum, die Einstellung der Entwickler bezüglich DPP kennenzulernen.

Die Entwicklung der Fragebogen fand anhand verschiedener Methoden und Hilfsmittel statt. Für Zielsetzung und Inhalte der Fragebogen war die wichtigste Methode die *Goal-Question-Metric* (kurz GQM) von Basili et al. [BW84, BCR94]. Zur Erstellung diente außerdem Literatur über die Erstellung von Fragebogen [Hol75, Sch93] und ein weiterer Fragebogen, der für vorherige Befragungen zum Thema PP von der AG SE eingesetzt wurde. Methoden und Hilfsmittel werden im Weiteren näher erläutert.

GQM

Die grundsätzliche Idee bei GQM ist, dass man sich bei der Planung nach einem Top-down-Prinzip an die zu erörternden Aspekte (Forschungsziele) annähert, dabei sein Ziel nicht aus den Augen verliert und Fragen in der Form stellt, dass gewünschte Messergebnisse erzielt werden können. Zur Auswertung der Daten wird später nach dem Bottom-Up-Prinzip vorgegangen, indem die gemessenen Daten verwendet werden, um die für das Forschungsziel gestellten Fragen zu beantworten. Es kann vorkommen, dass Maße später noch einmal angepasst werden müssen, wenn zwischenzeitlich feststellt wird, dass die gesetzten Forschungfragen dadurch noch nicht beantwortet werden.

Abbildung 3.2 zeigt das hierarchische Modell, das hinter GQM steht. Die oberste Ebene ist die konzeptionelle Ebene (G); hier werden Forschungsziele, die mit den Fragen erörtert werden sollen, definiert. Eines der Forschungsziele dieser Arbeit war bspw. zu bestimmen, wie verteilte Paarprogrammierung (DPP) aus Sicht der Entwickler erfolgreich eingesetzt werden kann.
Die mittlere Ebene (Q) ist die operationale Ebene. Hier werden Fragen formuliert, die sich auf eines der vorher definierten Ziele der obersten Ebene beziehen. Zum Beispiel war eine der Fragen: Welche Aspekte bringen den Erfolg von DPP-Sitzungen hervor?
Die unterste Ebene (M) ist die quantitative Ebene. Hier werden messbare Daten mit den vorher erstellten Fragen assoziiert. Dabei ist zu beachten, ob die Daten objektiv (z.B. Anzahl der Sitzungen) sind oder ob subjektive Abhängigkeiten (z.B. Zufriedenheit der Entwickler) bestehen. Der Fragebogen enthielt bspw. die Frage: Welche Vorteile soll die folgende Sitzung erbringen?

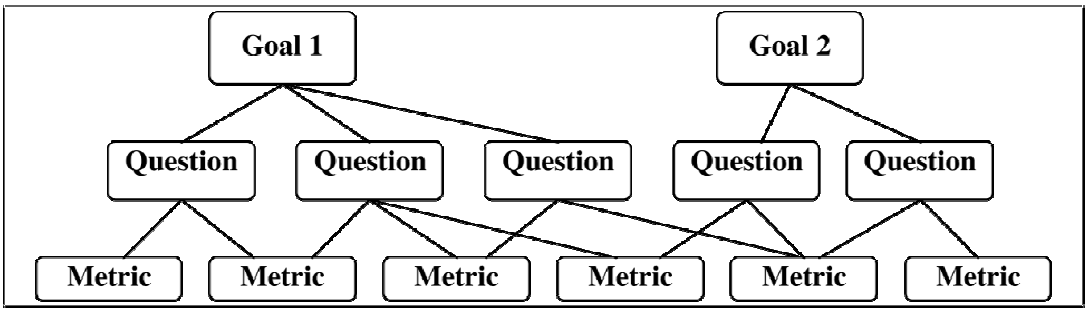


Abbildung 3.2 GQM-Modell nach Basili et al. [BCR94]

Mit Hilfe des GQM-Ansatzes lassen sich folgende Teilschritte zur Erstellung und Auswertung eines Fragebogens festlegen:

1. Ziele der Datenerhebung festlegen (G)
2. Fragen zu den Zielen formulieren (Q)
3. Messbare Datenkategorien erstellen (M)
4. Fragebogenkonzept erstellen (M)
5. Daten sammeln und validieren (M)
6. Daten analysieren (Q)
7. Daten auswerten (G)

Jeder dieser Schritte kann in das vorgestellte GQM-Modell eingeordnet werden. Die Planung zur Erstellung der Fragebogen ist in Anhang A einzusehen. Für die Definition der Forschungsziele im Fragebogen war bspw. das GQM-Template aufgeteilt in Objekt, Zweck, Fokus, Perspektive und Kontext, besonders hilfreich. Bei den Fragen zum Ziel durften auch Überlappungen auftreten. Es war in diesem Teil besonders wichtig, keinen Aspekt des Forschungsziels zu vergessen. Danach wurden die Fragen zu den zu messenden Daten auf drei verschiedene Fragebogen (Basisfragebogen, Fragebogen vor der Sitzung, Fragebogen nach der Sitzung) sinnvoll verteilt.

Fragebogen

Einmalig wurden am Anfang des DPP-Projektes im ersten Fragebogen (*General Questions*, kurz GQ) demographische Daten über die Entwickler, ihre Erfahrungen und ihre Einstellung bezüglich DPP erhoben.
Die weiteren Fragen wurden für jede Sitzung auf einen Fragebogen vor der Sitzung und einen Fragebogen nach der Sitzung aufgeteilt. Im Fragebogen vor der Sitzung (*Questionnaire Upcoming Session*, kurz US) wurden Einschätzungsfragen integriert, die sich auf Erwartungen der Entwickler bezüglich ihrer Ziele und DPP bezogen. Im Fragebogen nach der Sitzung (*Questionnaire Most Recent Session*, kurz MRS) wurden Bewertungsfragen eingefügt, die abfragen sollten, ob sich die Erwartungen der Entwickler erfüllt haben und ob ihre Ziele erreicht wurden. Vorlagen dieser Fragebogen befinden sich in Anhang B.
Die Fragebogen wurden mit dem Microsoft Word Formularassistent als Formular erzeugt. Um die Tauglichkeit der Fragebogen zu testen, ließ ich drei verschiedene Entwickler aus dem Saros-Team mit DPP-Erfahrung die Fragebogen probeweise ausfüllen. Dabei sollten folgende Punkte geprüft werden:

- wie lange das Ausfüllen der Fragebogen dauert
- mit welchem Programm der Fragebogen geöffnet wird
- ob das Layout gut ist (Kompatibilität mit Texteditoren)
- ob alle Fragen verständlich sind
- was noch verbesserungswürdig ist
- ob weitere Fehler vorliegen

Der *Pretest* (dt. Vorversuch) stellte sich als sehr sinnvoll heraus. Er zeigte, dass die Entwickler nicht mehr als insgesamt 15 Minuten für die Beantwortung aller Fragebogen brauchten. Auch die Verbesserungsvorschläge stellten sich als hilfreich heraus.

Vor- und Nachteile von Frageformen

Generell werden in der Literatur folgende Vor- und Nachteile mit Frageformen assoziiert [Hol75, Sch93].
Eine wichtige Überlegung bezog sich darauf, ob die Verwendung geschlossener oder offener Fragen vorzuziehen ist. Offene Fragen sind Fragen ohne Antwortvorgabe, im Gegensatz dazu geben geschlossene Fragen Antwortmöglichkeiten vor [Hol75].
Offene Fragen eignen sich bei Fragen zu Objekten, mit denen man sich noch nicht so gut auskennt. Man kann sich über die Antworten an die Vielfalt der Eigenschaften des Objektes annähern. Außerdem wird der Befragte nicht schon mit der Frage auf bestimmte Grenzen bei der Beantwortung eingeengt, so dass man evtl. mehr oder nuanciertere Informationen bekommt als erwartet. Es besteht jedoch das Risiko, dass der Befragte evtl. die Frage auf einer anderen Abstraktionsebene beantwortet, als man erwartet hatte, und die Antwort zu allgemein oder ungenau ist. Ein weiterer Nachteil von offenen Fragen ist, dass sie von dem Befragten erfordern, eine Antwort zu produzieren. Dies kann evtl. länger dauern als sich zwischen mehreren Antwortmöglichkeiten zu entscheiden, wie es bei geschlossenen Fragen der Fall ist. Bei der Auswertung der Antworten auf offene Fragen fehlt oft die Vergleichbarkeit, weil sie im Gegensatz zu geschlossenen Fragen nicht standardisiert sind.
Auch Antworten auf geschlossene Fragen können Verzerrungen durch die Struktur der Frage enthalten, z.B. wenn nur eine Antwortmöglichkeit in die Frage eingefügt wird. Beispielsweise könnte man fragen, ob der Entwickler DPP für eine sinnvolle Praktik zur Softwareentwicklung hält. In dieser Frage wäre dann nur eine Antwortmöglichkeit enthalten, im Gegensatz zu der Frage, ob der Entwickler DPP für eine sinnvolle Praktik hält oder nicht. Außerdem besteht die Gefahr, dass die Reihenfolge der Antwortvorgaben die Antwort beeinflussen könnte. Erstellt man längere Listen von Antwortmöglichkeiten, kommt es bei der Beantwortung der Fragen noch mehr auf die Reihenfolge der Antwortmöglichkeiten an. Es wurde bspw. nachgewiesen, dass obere und untere Elemente eine höhere Chance haben ausgewählt zu werden und dass das Ende des zweiten Drittels am wenigsten Aufmerksamkeit bekommt.

Soziale Einflüsse auf die Befragten war ein weiterer Aspekt, der die Formulierung der Fragen erschwerte. Man spricht von sozialer Wünschbarkeit von Antworten, wenn Befragte eine Antwort geben, die ihrer Meinung nach gesellschaftlich von ihnen erwartet wird [Hol75]. Mit anderen Worten werden Antworten, die gesellschaftlich sanktioniert sind, von den Befragten durch Konformitätsdruck und gesellschaftliche Erwartungen in eine bestimmte Richtung angepasst [Hus87]. Dies sind z.B. Fragen nach dem Einkommen oder der politischen Einstellung. Mit dem ersten Fragebogen (GQ) sollte bspw. erhoben werden, ob die Entwickler bisher lieber im Team oder alleine ihre Aufgaben gelöst haben. Diese Frage konnte aus Gründen der sozialen Wünschbarkeit nicht so direkt gestellt werden, da es sich bei dem Projekt ausschließlich um Teamarbeit handelte und wahrscheinlich keiner der Befragten darauf eine ehrliche Antwort gegeben hätte.
Die Auswertung der Fragebogen zeigte jedoch, dass trotz der Versuche, die beschriebenen Probleme bei Fragebogen zu umgehen, manche Antworten durch genau diese Probleme verzerrt wurden. Hierauf wird bei der Auswertung der Fragebogen (siehe Abschnitt 5.3.2) näher eingegangen.

3.2.2.2 Zielsetzung und Planung der mündlichen Befragung

Es gab zwei verschiedene Formen der mündlichen Befragung im Etablierungsprozess. Zum einen wurden Gruppengespräche in Reflexionen zu jeder Sitzung durchgeführt. Zum anderen stellte sich im Verlauf des Forschungsprozesses heraus, dass es zu bestimmten Gelegenheiten notwendig war, die Entwickler unabhängig voneinander in Einzelgesprächen zu befragen.

Reflexion

Die regelmäßigen Reflexionen zu den Sitzungen sollten allen Beteiligten die Möglichkeit des Informationsaustausches außerhalb der Sitzungen bieten. Vor allem sollten die Reflexionen dazu dienen, dass die Entwickler sich über Ereignisse, Erfahrungen und Probleme in den Sitzungen austauschen konnten. Die mündliche Form der Befragung sollte zusätzlich Aufschluss über die Wahrnehmung der Entwickler von Situationen oder Sachverhalten des Etablierungsprozesses erbringen.
Durch gezielte Fragen sollte das Gespräch in den Reflexionen auf kritische Punkte des Etablierungs- und Entwicklungsprozesses gelenkt werden. In diesem Zusammenhang bestand die Möglichkeit zu Rückfragen bezüglich der Beobachtungen oder der Angaben in den Fragebogen.

Es gab im Vorhinein einen teilstrukturierten Plan, wie der Ablauf der Reflexionen aussehen sollte (siehe Reflexionsleitfaden in Anhang C). Insgesamt sollte das Gespräch so mühelos und wenig unangenehm wie möglich für alle sein. Dabei wurde versucht, die folgenden Empfehlungen für mündliche Befragungen nach Holm [Hol75] zu berücksichtigen:

1. Einleitungsfrage stellen als Eisbrecher
    a. soll Misstrauen verringern und Antwortbereitschaft erhöhen
    b. Frage muss Befragten interessieren und leicht zu beantworten sein
2. Spielfragen zwischen schweren Fragen stellen
    a. Erholungsphase bzw. Puffer neutralisiert die Situation, sodass man unbeeinflusst zur nächsten Frage übergehen kann
3. Dramaturgie des guten Fragebogens
    a. ständiger Wechsel zwischen Spannung und Entspannung
    b. wechselnde Themen
    c. Fragetechnik wechseln
4. Dauer max. 40 Minuten wegen Konzentration
5. Abschiedsfrage: soll unverbindlich und anspruchslos sein, weil Befragter dann weniger das Gefühl hat, viel erzählt zu haben

Die vorher erwähnten Erkenntnisse über Vor- und Nachteile sowie geschickte Formulierung und Reihenfolge der Frageformen zur schriftlichen Befragung wurden auch für die mündliche Befragung verwendet.
Anfänglich war geplant, immer unstrukturierte und offene Fragen zu stellen, um die Antworten der Entwickler nicht zu sehr zu beeinflussen. Für den Fall, dass die Entwickler nicht auf die relevanten Punkte zu sprechen kommen würden, war angedacht, im Anschluss konkretere Fragen zu stellen.
Die mündliche Befragung in Form einer Gruppendiskussion zu führen, hat den Vorteil, dass ein gruppenweiter Lernprozess ermöglicht wird. Die Entwickler bekommen die Möglichkeit, wie im Forschungsprozess erwünscht (siehe Abschnitt 3.1 über CAR), aus den Gesprächen Erkenntnisse über die Erwartungen und Erfahrungen der anderen Beteiligten zu erhalten. Es kann das Verständnis füreinander fördern und die Lösung von wiederkehrenden Problemen vorantreiben.
Bei den Gruppengesprächen gibt es zwei erwähnenswerte Nachteile. Erstens besteht, wie bei den Fragebogen, die Gefahr des Konformitätsdrucks. Zweitens kann nicht ausgeschlossen werden, dass manche Teilnehmer in der Gruppe nicht zu Wort kommen, obwohl sie etwas zu sagen haben,

weil sie sich in Gruppengesprächen eher zurückhaltend verhalten oder andere Gesprächsteilnehmer ihnen zuvorkommen.

Einzelgespräch

Für die Umsetzung von Maßnahmen während des Etablierungsprozesses und für die abschließende Erhebung wurden mit den Entwicklern Einzelgespräche geführt. Dies hatte zum einen den Vorteil, dass ein Gespräch zu zweit mehr Vertraulichkeit und Direktheit besitzt. Zum anderen sprachen die Entwickler unabhängig voneinander und beeinflussten sich nicht gegenseitig. Als Beispiel für die Vorbereitung von Einzelgesprächen werden nun die Ziele und die Planung der abschließenden Erhebung beschrieben.

Mein Ziel bei der abschließenden Erhebung war von den beteiligten Entwicklern eine Bewertung in Hinsicht auf den Erfolg und Verbesserungsmöglichkeiten des Entwicklungs- und Etablierungsprozesses zu erhalten. Der Schwerpunkt der Befragung sollte auf der Erhöhung des Nutzens für das DPP-Projekt im Kontext des industriellen Einsatzes sein. Die Erhebung sollte sich nur auf die Perspektive der beteiligten Entwickler und ihre Erfahrungen beziehen. Dabei sollten auch konkrete Beobachten aus dem Etablierungsprozess in die Befragung einfließen, um problematische Aspekte noch einmal zur Sprache zu bringen.

Zur Planung diente wieder die GQM-Methode (siehe Anhang C). Danach stellte sich die Frage, auf welche Art die Erhebung am besten durchführbar wäre, um die gewünschten Informationen am ehesten zu erhalten. Zur Wahl standen Fragebogen, eine Gruppendiskussion (wie in den Reflexionen) oder ein Interview.
Die Erfahrungen aus dem bisherigen Forschungsprozess halfen diese Entscheidung zu treffen. Die Gruppendiskussion hätte das Risiko beinhaltet, dass sich die Antworten der Befragten gegenseitig beeinflussten. Der Fragebogen hätte gegenüber dem Interview den Vorteil gehabt, dass die Entwickler sich evtl. mehr Zeit, als in einem Gespräch, genommen hätten, um die Fragen zu beantworten.
Die Reflexionen hatten gezeigt, dass bei Interviews das Risiko von unerwarteten Gesprächsverläufen und spontanen Rückfragen seitens der Entwickler bestand. Einer der Entwickler hatte bspw. bei einem Gespräch über Kommunikation in Frage gestellt, ob Kommunikation außerhalb der Sitzungen nötig wäre, was vorher als eine Selbstverständlichkeit vorausgesetzt wurde. Schließlich war das Risiko von unerwarteten Gesprächsverläufen und spontanen Rückfragen seitens der Befragten kleiner, als nicht genug Informationen durch die Befragung zu erhalten.
Die Wahl fiel am Ende auf ein teilstrukturiertes Interview. Zum einen hatte es den Vorteil, dass Rückfragen gestellt werden konnten, wenn Antworten unklar oder zu wenig detailliert beantwortet wurden. Es war bei Fragebogen zur Sitzung manchmal vorgekommen, dass die Entwickler zu unpräzise Antworten gaben oder Begründungen ausließen. Zum anderen beinhaltete die Methode des teilstrukturierten Interviews, dass die Entwickler die angesprochenen Themen vorrangig selbst wählen konnten.

Der Aufbau der Fragen war wie im Interviewleitfaden beschrieben (siehe Anhang C). Als Einstieg sollte eine offene Frage, die leicht zu beantworten war, einen ersten Hinweis darüber geben, welche Aspekte den Entwicklern bei DPP besonders wichtig waren oder was ihnen spontan dazu einfiel.
Ein teilstrukturiertes Interview hat in der Regel keinen festen Ablauf. Es gibt Schwerpunkte, die in offenen Fragen angesprochen werden. Doch es ist dem Befragten durch seine Antwort überlassen, in welche Richtung sich das Gespräch entwickelt. Der Interviewer greift vom Befragten erwähnte Themen auf, um die Befragung weiterzuführen.
In den Reflexionen hatte ich die Erfahrung gemacht, dass die Entwickler nur so viel wie unbedingt nötig von selbst erzählten. Daher war ich darauf vorbereitet, dass sie nicht ausführlich auf die erste Frage antworten würden. Die weiteren Fragen waren daher konkreter und bezogen sich darauf, herauszufinden, welche Ziele die Entwickler hatten, wie sie den Erfolg über ihre Ziele

bemessen hatten und was für ein Fazit sie für sich aus den Erfahrungen mit DPP zogen. Abschließend hatte ich zur Sicherheit noch einige kritische Punkte bzgl. der Effizienz in den Sitzungen notiert, die ich verwenden konnte, falls das Gespräch über die Ziele ins Stocken kommen sollte.
Eine Beschreibung des Verlaufs und der Ergebnisse befindet sich in Abschnitt 5.1.

3.3 Verfahren der Datensicherung

3.3.1 Beobachtungsprotokoll

Zuerst versuchte ich, den chronologischen Verlauf der Sitzung ohne Vorlage zu protokollieren. Später unterteilte ich die Geschehnisse in organisatorische, inhaltliche und technische Aspekte. Doch es stellte sich heraus, dass in den Aufzeichnungen manche grundsätzlichen Details zur Sitzung, z.B. beteiligte Entwickler, Sitzungsinhalte etc. und eine strukturierte Vorgehensweise fehlte.
Im Verlauf des Projektes und zur späteren Analyse verwendete ich daher eine Vorlage, die weitere Details berücksichtigte. Zuerst wurden allgemeine Details über die Sitzung abgefragt, z.B. Sitzungsdatum und beteiligte Entwickler. Danach folgte ein Abschnitt für die chronologische Auflistung von Geschehnissen inklusive zusätzlicher Felder wie z.B. die aktuelle Uhrzeit. Daraus entstand ein Protokoll, das die aufgezeichneten Vorkommnisse strukturiert darstellte.

3.3.2 Technische Infrastruktur zur Beobachtung und Aufzeichnung

Anfänglich wurden die Beobachtungen und die Aufzeichnungen aus der Perspektive eines weiteren Entwicklers durchgeführt. Ich teilte wie ein weiterer Entwickler die Editorfenster des Projektes in Saros. Die Bildschirmanzeige wurde mit Hilfe der Videoaufzeichnungssoftware Camtasia [Camtasia] aufgenommen. Zur besseren Übersicht wurde die Aufzeichnung auf einen größeren externen Bildschirm mit einer höheren Auflösung ausgelagert. Die Sitzungen wurden ohne Videoübertragung durchgeführt. Alle weiteren Details und eine Abbildung zur technischen Infrastruktur befinden sich in Kapitel 4.3.

Im Verlauf des Projektes stellte sich jedoch heraus, dass es aus verschiedenen Gründen von Vorteil war, die Beobachtungen über eine Desktop Sharing-Anwendung [Netviewer] durchzuführen. Ab diesem Zeitpunkt wurde dann auch die Netviewer-Sitzung zur Aufzeichnung der Sitzung verwendet.
Einer der Vorteile der Netviewer-Sitzung war, dass zusätzlich zu Saros noch das Verhalten von einem der Entwickler in seinem Workspace beobachten werden konnte. Dadurch wurde bspw. deutlich, wann dieser sich Dateien und Anwendungen außerhalb von Eclipse anschaute.
Ein weiterer Vorteil war, dass in Fällen, in denen Saros nicht funktionierte, es trotzdem möglich war, Einblick in die Aktivitäten der Entwickler zu haben. Besonders als in einer der Sitzungen, in der Saros aus technischen Gründen nicht verwendet werden konnte, die Entwickler anhand von Spezifikationsdokumenten ihre Anforderungen besprachen.

Die Audiokommunikation wurde im Projekt anhand einer kostenlosen VoIP-Software für Audiokonferenzen [Mumble] eingerichtet und mit Hilfe einer Software zur Signalweiterleitung Virtual Audio Cable [VAC] an die Camtasia-Aufzeichnung übertragen.
Die Aufzeichnung der Kommunikation wurde zuerst in einer Zwei-Rechner-Lösung umgesetzt. Dafür wurde auf zwei verschiedenen Rechnern ein Mumble Client eingerichtet. Der erste Rechner (Mumble Recorder), auf dem auch die Videoaufzeichnung lief, zeichnete die Audiosignale auf. Der Mumble Client auf dem zweiten Rechner wurde verwendet, um mit den anderen Teilnehmern zu kommunizieren.
In der später entwickelten Ein-Rechner-Lösung wurde die technische Infrastruktur zur Aufzeichnung inklusive des Audio-Recorders auf demselben Rechner in einer virtuellen Maschine [VirtualBox] ausgeführt.

3.4 Verfahren der Datenauswertung

In Forschungsgebieten, bei denen soziale Strukturen und Interaktionen eine wichtige Rolle spielen, kann die Komplexität der Einflussfaktoren unbeschreiblich groß werden. Unter diesen Bedingungen bietet sich die Auswertung anhand von qualitativen Forschungsmethoden an, um auftretende Phänomene zu beschreiben und Erkenntnisse daraus zu ziehen. Diese Methoden ermöglichen eine systematische Aufbereitung der Daten und Erkenntnisse, ohne dabei den Raum für Interpretationen und die Vielschichtigkeit der Geschehnisse zu sehr einzuengen.

Die grundlegende Forschungsmethode (CAR) sieht vor, dass innerhalb der Zyklen des Forschungsprozesses sich Datenerhebung und -auswertung immer wieder abwechseln. Daher ist die Analyse der Daten schon während des Etablierungsprozesses ein wichtiger Bestandteil. Im allgemeinen Prozessmodell (siehe Abbildung 3.1) werden die Teilschritte als Diagnose und Auswertung aufgeführt. Im Etablierungsprozess (siehe Abbildung 4.3) waren sie Teile der Ist- und Soll-Analyse.

Nach Abschluss der Datenerhebung wurde zur vertiefenden Analyse der Daten eine Variante der qualitativen Inhaltsanalyse verwendet, die auf interpretativen Verfahren [Brü00, Ric05, Hus87] beruht. Als Orientierung diente dabei bspw. Grounded Theory (kurz GT). Da GT unter anderem eine strukturierte Codierung der Daten und eine Analyse, die sehr nah an den Daten ist, ermöglicht [SC90].
Für die Analyse waren verschiedene Schritte notwendig. Zuerst wurden die aufgezeichneten Daten aus Beobachtungen und Befragungen nach verschiedenen Kategorien sortiert. Diese Kategorien enthielten Unterkategorien, sogenannte Konzepte, nach denen alle Phänomene eingeordnet waren. Der Begriff Phänomen bezeichnet ein Ereignis, eine Aussage oder eine Angabe im Fragebogen innerhalb des Etablierungsprozesses. Im nächsten Schritt wurden unter der Vielzahl von Konzepten einige herausgesucht und diese genauer analysiert und mit weiteren Daten mit ähnlichen Eigenschaften in Verbindung gebracht.
Die Beschreibung der Phänomene (siehe Abschnitt 5.2) erfolgte anhand von konkreten Daten aus dem Etablierungsprozess. Dafür wurden sie in den Kontext der Sitzungen und des gesamten DPP-Projektes eingebettet. Aus der Analyse ließen sich schließlich weitere Erkenntnisse über den Etablierungsprozess und DPP im Allgemeinen ableiten (siehe Abschnitte 5.3 und 5.5).

4 Beschreibung des Etablierungsprozesses

In diesem Kapitel werden alle Hintergrundinformationen über den Etablierungsprozess vorgestellt. Dies beinhaltet die Beteiligten Kooperationspartner, die beteiligten Entwickler und weitere vorbereitende Maßnahmen. Weiterhin werden auch alle Voraussetzungen wie z.B. die technische Infrastruktur und weitere Informationen über die Entwickler und den Entwicklungsprozess erläutert. Schließlich folgt eine Beschreibung des Etablierungsprozesses inklusive der Erläuterung des Aufbaus, der Vorgehensweise und einer chronologischen Übersicht über die Ereignisse.

4.1 Projektbeteiligte Kooperationspartner

In diesem Abschnitt werden kurz die Kooperationspartner für den Etablierungsprozess vorgestellt. Zum einen war das beteiligte Unternehmen, Teles, und das Saros-Team der AG SE. Die beteiligten Entwickler werden in einem späteren Abschnitt (siehe Abschnitt 4.4) vorgestellt.

4.1.1 Das Unternehmen

Die Teles AG Informationstechnologien (kurz Teles) [Teles] ist ein mittelständisches Unternehmen in der Informationstechnologiebranche mit einem jährlichen Umsatz von 23,8 Mio. Euro [Tel08]. Teles wurde 1983 als GmbH in Deutschland gegründet. 1996 wurde das Unternehmen in eine Aktiengesellschaft umgewandelt. Teles vertreibt seit 26 Jahren Telekommunikationsprodukte. Das Unternehmen hat sich als Anbieter für Ausrüstungen, Lösungen und Dienstleistungen für Festnetz- und *Fixed-Mobile Convergence* sowie für *Next-Generation-Network* (NGN) Dienstleister auf dem Markt etabliert. Der Kundenstamm umfasst mehr als 200 Netzwerkbetreiber. Im Bereich von VoIP-Lösungen und speziell für NGN/IMS Technologien der Class 4 und Class 5 NGN-Lösungen ist Teles besonders innovativ. Zu den Kunden von Teles zählen Unternehmen wie z.B. Freenet, T-Mobile und T-Systems. Der Gründer von Teles, Prof. Dr. Ing. Sigram Schindler, war bis 2001 als Professor an der Technischen Universität Berlin tätig. Er ist heute Hauptaktionär und Vorstandschef des Unternehmens. Das Unternehmen ist international tätig und hat seinen Hauptsitz in Berlin. Weitere Niederlassungen befinden sich in Wien (Österreich), Genua (Italien), Yokne'am (Israel), Bangalore (Indien), Sao Paulo (Brasilien) und Santa Cruz (Bolivien). Des Weiteren existieren Kooperationen mit internationalen Telekommunikationsunternehmen aus mehr als 20 Ländern.

4.1.2 Das Saros-Team

Das Saros-Team wurde 2006 im Rahmen der ersten Diplomarbeit zur Entwicklung von Saros gegründet. Es bestand zum Zeitpunkt des DPP-Projektes aus dem projektverantwortlichen Professor, einem wissenschaftlichen Mitarbeiter, der die technische Koordination der Saros-Softwareentwicklung leitete, und einem wissenschaftlichen Mitarbeiter, der für die Koordination des DPP-Projektes zuständig war. Außerdem waren durchgängig mindestens vier weitere Studenten im Rahmen wissenschaftlicher Arbeiten mit der Weiterentwicklung von Saros beschäftigt.

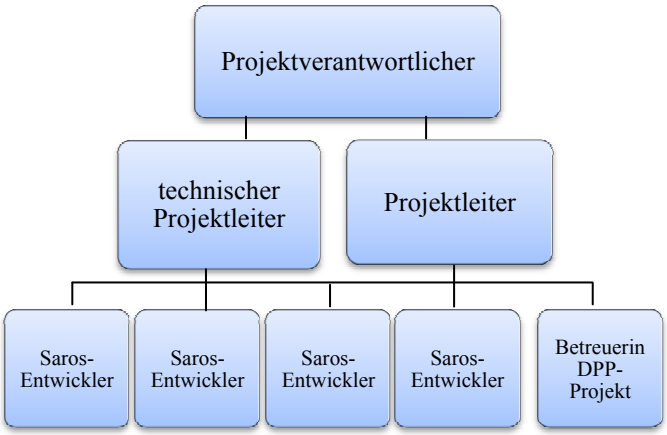


Abbildung 4.1 Das Saros-Team und die Positionen der Teammitglieder

4.2 Planungsphase

Alles was für den Etablierungsprozess vorbereitet werden konnte, wurde in der Planungsphase erledigt. Als ich in das DPP-Projekt einstieg, stand das Saros-Team bereits in Kontakt mit den Projektverantwortlichen der Teles AG. In der Kommunikation ging es hauptsächlich um die Klärung von technischen Voraussetzungen für die Sitzungen und um die Vorbereitung einer ersten Demonstrationssitzung von Saros im Unternehmen. Es stand bereits fest, dass die DPP-Sitzungen zwischen der C5-Abteilung in Bangalore und Wien (siehe Abschnitt 4.4 über Details der C5-Abteilung) etabliert werden sollten und dass es sich dabei um ein Java-Projekt handeln würde. Alle weiteren Details wie z.B. Sitzungsziele, -dauer oder –inhalte waren dagegen noch offen (siehe Abschnitt 4.6 über den Ablauf des Etablierungsprozesses).
Zu den technischen Voraussetzungen, die noch geklärt werden mussten, gehörten z.B. welche Hardware (Bildschirme, Headsets, etc.) eingesetzt werden sollte, ob die verfügbare Bandbreite zwischen den Büros in Wien und Bangalore für DPP-Sitzungen ausreichen würde oder welche Audio Software verwendet werden sollte.
Es vergingen weitere zwei Monate zwischen dem ersten Demonstrationstermin von Saros bei Teles in Berlin und der ersten Pilotsitzung. Saros war zu diesem Zeitpunkt noch in der α-Version und die Zeit wurde vom Saros-Team genutzt, um die Software in einen Zustand zu bringen, der für die Durchführung von DPP-Sitzungen notwendig war. [2]
Zur Vorbereitung auf die Etablierung mussten die einzelnen Komponenten der technischen Infrastruktur ausgewählt und gemeinsam getestet werden (siehe Abschnitt 3.3 und 4.3).
Bei der Demonstrationssitzung lernte ich einen an den DPP-Sitzungen beteiligten Entwickler (weiterhin als Entwickler1 bezeichnet) kennen. Ich nutzte das Treffen, um mehr über die Ziele für den Einsatz von DPP zu erfahren. Er teilte mir mit, dass eines der zentralen Ziele sei, dass sich durch die DPP-Sitzungen alle beteiligten Entwickler im Code auskennen würden und zukünftig Implementierung und Wartung übernehmen könnten (siehe auch Abschnitt 4.4.2 zu Zielen der Entwickler).
Unabhängig davon recherchierte ich über das Gebiet der Paarprogrammierung im Allgemeinen und DPP im Besonderen (siehe Kapitel 2). Weiterhin versuchte ich, mich auf mögliche Beobachtungsinteressen innerhalb der Sitzungen vorzubereiten, indem ich untersuchte (1) welche Kriterien eine Sitzung erfolgreich bzw. nützlich machen, (2) was für potentielle Problemquellen es geben könnte und (3) inwiefern sich die Sitzungssituation von bereits erforschten Formen der DPP unterscheidet. Beispielsweise recherchierte ich anhand der Informationen aus dem Treffen mit Entwickler1 über mögliche Sitzungstypen (siehe Abschnitt 2.3 über Sitzungstypen in PP).
Die Recherche zur Forschungsmethode sollte mir Wege aufzeigen, den Etablierungsprozess effizient zu gestalten sowie die Daten sinnvoll zu erheben und zu verwerten. Außerdem versuchte ich

[2] Vgl. Diplomarbeit eines Saros-Entwicklers aus dieser Zeit [Jac09].

soweit wie möglich zu ermitteln, was der Inhalt und der Umfang (*Intension* und *Extension* nach [Hus87]) der Arbeit sein sollte.
Ich wählte passende Hilfsmittel für den Etablierungsprozess wie z.B. Fragebogen und Leitfäden, gestaltete ihr Layout und verfeinerte sie anhand der gesammelten Informationen.

4.3 Technische Infrastruktur der DPP-Sitzungen

Für die DPP-Sitzungen wurde neue Hardware angeschafft und eingerichtet. Dazu gehörten bspw. ein zusätzlicher Rechner von Teles, auf dem alle notwendigen Server betrieben wurden, und USB-Headsets für alle Beteiligten.
An den Sitzungen nahmen mindestens drei Personen verteilt auf drei Standorte (Bangalore, Berlin und Wien) teil, die bis auf wenige Ausnahmen alle dieselbe Software installiert hatten (siehe Tabelle 4.1). Von den Beteiligten wurden Windows XP und Vista als Betriebssysteme verwendet. Bei jedem der Teilnehmer wurde die kostenlose VoIP-Software für Audiokonferenzen Mumble zur Audiokommunikation eingerichtet. Weiterhin hatten die Entwickler zur Softwareentwicklung die Entwicklungsumgebung Eclipse mit dem Saros-Plugin eingerichtet. Einer der Entwickler teilte per Desktop- Sharing mittels Netviewer Meet seinen Desktop mit dem Rechner, über den die Sitzungen aufgezeichnet wurden (siehe Abschnitt 3.3).
Für die Sitzungen bei Teles gab es zwei Kommunikationsserver. Die Kommunikation zwischen allen Beteiligten fand über das Internet statt. Um die Firmengeheimnisse zu wahren, wurden aus Sicherheitsgründen die Kommunikationsserver auf dem zusätzlichen Rechner bei Teles eingerichtet. Sie waren nur über einen speziell für die Sitzungen eingerichteten VPN-Tunnel [OpenVPN] zum Firmennetzwerk erreichbar.
Alle Informationen zur technischen Infrastruktur, die zum Aufbau einer DPP-Sitzung notwendig sind, werden auch auf der Homepage von Saros detailliert aufgelistet.

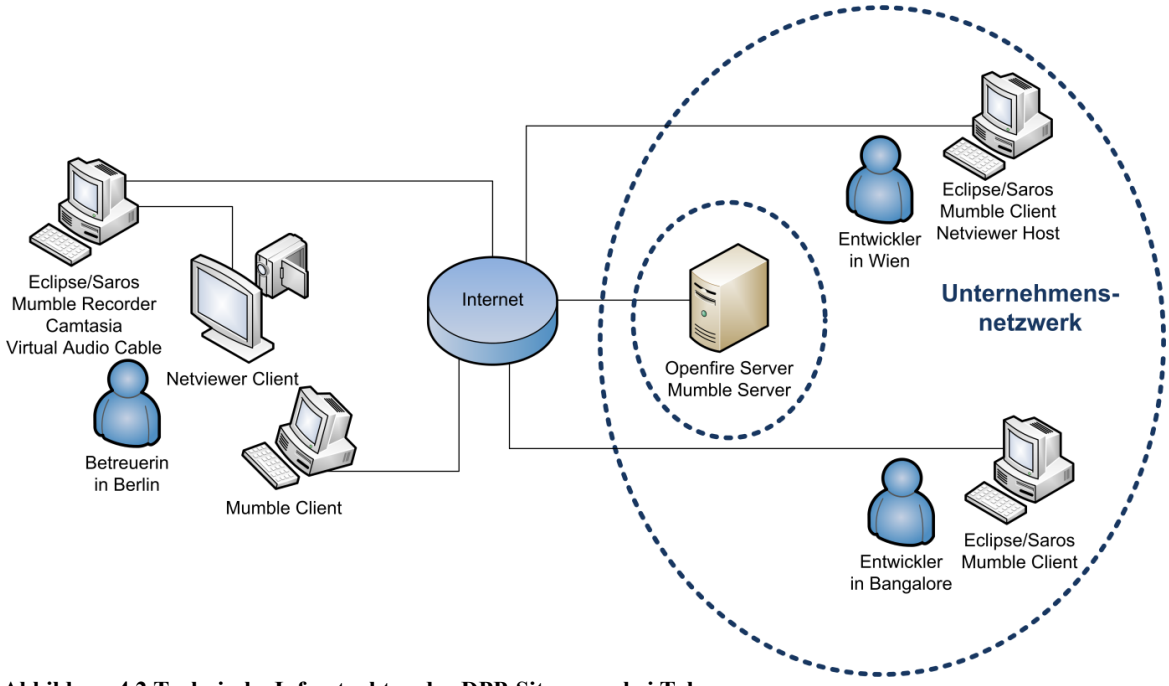


Abbildung 4.2 Technische Infrastruktur der DPP-Sitzungen bei Teles

Software	Verwendungszweck
OpenVPN	separates Netzwerk des Unternehmens für die Server (Mumble, Openfire)
Mumble	Audiokommunikation (Server und Clients)
Openfire	Jabber/XMPP-Server für Saros
Eclipse	Softwareentwicklungsumgebung
Saros	Eclipse-Plugin zur kollaborativen Softwareentwicklung
Weitere Software zur Beobachtung und Aufzeichnung	
Netviewer Meet	Desktop Sharing zur Beobachtung
Camtasia	Videoaufzeichnung zur Datensicherung
Virtual Audio Cable	Audiosignalweiterleitung zur Audiodatensicherung

Tabelle 4.1 Verwendete Software für die technischen Infrastruktur der DPP-Sitzungen im Etablierungsprozess

4.4 Beteiligte Entwickler

An den DPP-Sitzungen waren drei Java-Softwareentwickler von Teles beteiligt. Sie waren von Teles für das Projekt ausgewählt worden, weil sich die Zusammenarbeit zwischen Wien und Bangalore gerade im Aufbau befand. Entwickler1 arbeitete für die Forschungs- und Entwicklungsabteilung (kurz F&E) im Class 5-Team (kurz C5-Team) in Wien. Entwickler2 und Entwickler3 arbeiteten für das C5-Team der F&E-Abteilung in Bangalore.
Alle Entwickler hatten mehr als 3 Jahre Berufserfahrung. Entwickler1 hatte als projekterfahrener Entwickler am meisten Domänenwissen. Er hatte außerdem während der Studienzeit erste Erfahrungen mit PP als Softwareentwicklungsmethode gesammelt (siehe Tabelle 4.2).

Die Entwickler kannten sich vor Beginn der Sitzungen kaum. Die beiden Entwickler aus Bangalore hatten erst kurz vorher ihre Arbeit bei Teles angenommen (siehe Tabelle 4.2) und saßen seitdem im selben Büro nebeneinander. Entwickler1 hatte sie etwa zur selben Zeit bei einem einwöchigen Meeting in Wien persönlich kennengelernt. Er hatte ihnen in dieser Woche existierende Artefakte des C5-Teams gezeigt.

Beteiligte	Entwickler1	Entwickler2	Entwickler3
Abteilung	C5-Team Wien	C5-Team Bangalore	C5-Team Bangalore
Berufserfahrung	3 Jahre und 6 Monate	4 Jahre und 7 Monate	3 Jahre und 10 Monate
Teles-Mitarbeiter seit	1 Jahr und 4 Monaten	5 Monaten	5 Monaten
Domänenerfahrung	viel	wenig	wenig
PP-Erfahrung	aus der Studienzeit	keine	keine

Tabelle 4.2 Persönliche Informationen über am DPP-Projekt beteiligte Teles-Entwickler

4.4.1 Erfahrungen, Erwartungen und Motivation der Entwickler

Zu Beginn der Sitzungen wurden anhand eines Basisfragebogens (siehe General Questions in Anhang B) die Erfahrungen und Erwartungen der beteiligten Entwickler bezüglich DPP im Allgemeinen erhoben.
Dabei stellte sich heraus, dass die Entwickler aus Bangalore vor Beginn des Projektes nie etwas mit PP oder DPP zu tun gehabt hatten. Entwickler1 hatte erste Erfahrungen mit PP (nicht aber mit DPP) aus der Zeit seines Studiums gesammelt. Während seiner über dreijährigen Berufspraxis als Entwickler wurde es jedoch nicht eingesetzt.

Zu der Frage, was die Entwickler generell über DPP denken, gab es unterschiedliche Antworten (siehe Tabelle 4.3).

Entwickler1	*"I think that it can be used to exchange information and knowledge really good. Sometimes it might be useful to discuss a problem or search a bug together. I do not think that it really helps for straight forward programming tasks since it leads to overhead and can slowdown if the programmers have a very different skill or output level."*
Entwickler2	*"Two or more persons working together"*
Entwickler3	*"Well, according to me, it's a great way to make something done. But it's also true that nobody can feel the real benefit of using this unless used once. So I hope that this would help a lot and add a great amount of value to the overall development process."*

Tabelle 4.3 Angaben der Entwickler im Fragebogen General Questions darüber, was sie generell über DPP denken

Entwickler1 hatte bereits eine detaillierte Vorstellung davon, was er sich von DPP für die Sitzungen erhoffte. Er gab an, dass sich DPP eigne, um Informationen auszutauschen, Wissen zu transferieren, Probleme zu diskutieren und Defekte zu suchen. Gleichzeitig bezweifelte er jedoch, dass kollaborative Programmierung bzw. PP bei der Erzeugung von Code (*„straight forward programming"*) hilfreich sein würde, weil es zeitaufwendig wäre bzw. im Falle von unterschiedlichen Erfahrungsniveaus zu Zeitverzögerungen führen würde.
Entwickler2 äußerte keine konkreten Erwartungen bzgl. DPP. Er antwortete auf diese Frage, indem er ganz allgemein angab, dass bei kollaborativer Programmierung bzw. PP zwei oder mehr Personen zusammenarbeiten.
Entwickler 3 hatte keine konkreten Erwartungen an DPP. Er hoffte ganz allgemein, dass es eine Bereicherung für den vorhandenen Entwicklungsprozesses sein würde.

Die beteiligten Entwickler hatten sich nicht freiwillig für die DPP-Sitzungen gemeldet, sondern waren dafür ausgewählt worden. Es ist daher relevant, kurz ihre Motivation für die DPP-Sitzungen zu beschreiben. Ganz allgemein war bei allen Entwicklern ausreichend Motivation für die Durchführung der Sitzungen vorhanden. Die Entwickler bereiteten sich alle nur sehr kurzfristig auf die Sitzungen vor, und es kam bei jedem vor, dass er die Beantwortung von Fragebogen ausließ.
Es zeigten sich jedoch Nuancen zwischen der Motivation der Entwickler in der Unterstützung und in ihrem Interesse für den Etablierungsprozess. Entwickler3 schien sehr motiviert. Er antwortete immer zügig auf Emails, fragte nach, was von ihm erwartet wurde, und machte mehrere Angaben in den Fragebogen, die seine Motivation zeigten:*„It's nice to be part of this effort"* oder *„Well, according to me, it's a great way to make something done."*. Entwickler1 reagierte etwas langsamer auf Emails, dafür zeigte er sein Interesse durch sein Bestreben, verschiedene Sitzungstypen, z.B. *"Discover how debugging works."* ausprobieren zu wollen und unterstützte den Etablierungsprozess, indem er dafür sorgte, dass ich alle notwendigen Informationen bekam. Über die Motivation von Entwickler2 kann hier nicht viel ausgesagt werden. Er hielt sich mit persönlichen Aussagen zurück (siehe auch Tabelle 4.2). Seine Angaben im Fragebogen waren meistens kurz und sehr allgemein formuliert.

4.4.2 Ziele der Entwickler

In der Planungsphase wurde bereits erwähnt, dass eines der Projektziele der Entwickler war, dass durch DPP jeder von ihnen in der Lage sein sollte den Code später zu verändern. Eine meiner Aufgaben im Etablierungsprozess war dafür zu sorgen, dass die Teles-Entwickler die Sitzungen als für sich erfolgreich empfanden. Dafür war es nötig zu erheben, welche Ziele und Erwartungen die Entwickler an die einzelnen DPP-Sitzung hatten.

Um die Ziele der Entwickler für die DPP-Sitzungen zu ermitteln und problematische Situationen (im Sinne von erfolglosen Sitzungen) zu erkennen, sollten die Entwickler Angaben darüber machen, was ihr Verständnis einer erfolgreichen oder einer erfolglosen Sitzung sei. Im Basisfragebogen wurden sie einmal allgemein (siehe General Questions im Anhang) und später für jede DPP-Sitzung anhand von Fragebogen vor und nach der Sitzung (siehe Fragebogen Upcoming Session und Most Recent Session im Anhang) einzeln danach gefragt. Tabelle 4.4 und Tabelle 4.5 enthalten die Antworten der Entwickler aus dem Basisfragebogen. Einzelne Sitzungsziele werden im Abschnitt 4.7 bei der Beschreibung der Sitzungen erwähnt.

Entwickler1	*"If every participant thinks that he has obtained knowledge from other developers."*
Entwickler2	*„If we get a good quality code in the end"*
Entwickler3	*"To me, if we can concentrate on the real goal of the collaborative software development then we can see good results of using it."*

Tabelle 4.4 Angaben der Entwickler im Fragebogen General Questions über die Kriterien einer erfolgreichen Sitzung

Entwickler1	*"No information transfer or no code output"*
Entwickler2	*"if no output, and everybody is sticking to his/her view"*
Entwickler3	*"If we deviate from the track then it would result as a failure. So it's better to do the periodic review of the path taken."*

Tabelle 4.5 Angaben der Entwickler im Fragebogen General Questions über die Kriterien einer erfolglosen Sitzung

Aus der Perspektive von Entwickler1 ist eine Sitzung erfolgreich, wenn alle Beteiligten denken, dass sie von den anderen Wissen erhalten haben und mindestens Informationstransfer oder Erzeugung von Codeoutput daraus resultierte.
Entwickler2 hatte eine eher produktorientierte Sicht und erhob den Anspruch, in einer erfolgreichen Sitzung Code mit guter Qualität zu erzeugen. In einem abschließenden Einzelgespräch ergänzte er seine Aussage über die Qualität auf die Lauffähigkeit des Codes.
Für Entwickler3 resultierte eine erfolgreiche Sitzung aus einer zielorientierten Zusammenarbeit.

Projektziel	Alle Beteiligten kennen sich im Code aus und können die Implementierung und die Wartung übernehmen.
Entwickler1	Wissenstransfer und Code Erzeugung
Entwickler2	Code mit guter Qualität bzw. lauffähigen Code erzeugen
Entwickler3	zielorientierte Zusammenarbeit

Tabelle 4.6 Allgemeine Ziele des Teams und der einzelnen Entwickler im DDP-Projekt

Es ist zu diesem Zeitpunkt schon ersichtlich gewesen, dass die Entwickler grundsätzlich unterschiedliche Vorstellungen einer erfolgreichen bzw. weniger erfolgreichen Sitzung hatten. Die genannten Ziele wurden als erste Hinweise für mögliche Problemquellen in den Sitzungen aufgefasst und in der Analyse in Kapitel 5 weiter erörtert.

4.4.3 Rollen im Entwicklungsprozess

Die Rollen und Verantwortlichkeiten der Entwickler bestimmten sich anhand der Projekterfahrung. Es stellte sich im Verlauf der Sitzungen heraus, dass Entwickler1 in den Sitzungen eine leitende Rolle hatte. Er hatte als projekterfahrener Entwickler am meisten Domänenwissen. Außerdem wurde ihm von Teles die Verantwortung für die DPP-Sitzungen und deren Inhalt übertragen.
Entwickler2 und Entwickler3 hatten einige Monate mehr Berufserfahrung als Entwickler1, ihnen fehlte jedoch noch Wissen über die Domäne, das sie sich unter anderem mit Hilfe der Sitzungen

aneignen wollten. Sie wurden für die Sitzungen ausgewählt, da sie die beiden Entwickler in der Niederlassung in Bangalore waren, die mit Java programmieren konnten.
Entwickler1 war an allen Sitzungen beteiligt. Entwickler2 und Entwickler3 wechselten sich ab oder waren beide anwesend.
Von Williams und Kessler [WK02] wird diese Rollenverteilung in PP-Sitzungen als *Expert-Novice* bezeichnet (siehe Abschnitt 2.3.1).

4.4.4 Aufgabenverteilung im Entwicklungsprozess

Zur Erläuterung der Aufgabenverteilung und der Vorgehensweise in den Sitzungen teilte mir Entwickler1folgende Informationen über den Standardentwicklungsprozess von Teles mit:
Das Teles *Quality Management System* (dt. Qualitätsmanagementsystem, kurz QMS) legt neben anderen Abläufen im Unternehmen auch die Rahmenbedingungen für den Softwareentwicklungsprozess fest. Das Managementsystem des Unternehmens ist seit mehr als 10 Jahren ISO-zertifiziert, und die vorhandenen Qualitätsmaßstäbe entsprechen der aktuellsten Norm DIN EN ISO 9001:2008. Darin wird unter anderem festgelegt, welche Meilensteine es gibt, wer für die Planung und Ausführung verantwortlich ist und welche Dokumentation dabei mindestens enthalten sein soll.
Das C5-Team entwickelte auf Grundlage dessen einen Entwicklungsprozess, der auf ihre Anforderungen angepasst ist. Der Entwicklungsprozess besteht aus folgenden Phasen:

1. Spezifikation
2. Implementierung
3. Verifikation
4. Release

Diese finden in einer sequentiellen Reihenfolge statt. Die chronologische Vorgehensweise entspricht daher am ehesten einem Wasserfallmodell [Boe76]. Im Laufe des Entwicklungsprozesses finden wöchentliche C5-Statusmeetings statt, bei denen jedes Teammitglied den Status seiner Aufgaben der letzen und kommenden Woche sowie aufkommende Probleme darlegt.
In der Spezifikationsphase werden die Spezifikationsdokumente vom Produktmanager (*Product Line Manager*, kurz PLM) erstellt. Darin werden alle Anforderungen aufgelistet und beschrieben.
Anschließend werden die Aufgaben entsprechend den Anforderungen vom Leiter der C5-Abteilung (*Head of Development*, gleichzeitig Leiter der Abteilung F&E) den Entwicklern zugewiesen.
Die Anforderungen werden dann von den Entwicklern entsprechend ihren Aufgaben mit Hilfe von funktionalen und technischen Spezifikationsdokumenten (*Functional Application Document*, kurz FAD und *Technical Application Document*, kurz TAD) ausgearbeitet. Die Spezifikationsphase endet mit einem Review-Meeting über die FAD und TAD mit allen Stakeholdern, z.B. auch Entwicklern und Testmanagern.
In der Implementierungsphase werden die Aufgaben von den Entwicklern in unabhängigen Modulen bearbeitet und sobald sie fertig sind in die Gesamtlösung integriert.
Die in der Implementierungsphase erzeugten Artefakte werden in der Verifikationsphase dem Qualitätsmanagement (kurz QM) übergeben. Das QM-Team erzeugt anhand der Spezifikationsdokumente die Testdokumente (*Standard Test Description*, kurz STD). In einem weiteren Review-Meeting werden auch diese Dokumente überprüft und zur Verwendung freigegeben. Anhand der STD werden die Artefakte für das Release freigegeben. Mit dem Release ist die Verifikationsphase abgeschlossen.
Es gibt etwa zwei Mal pro Jahr ein Release. Dies ist die letzte Phase im Entwicklungsprozess. Hierin ist auch die Wartung der entwickelten Software enthalten.

Die Aufgabenverteilung wurde entsprechend des aktuellen Entwicklungsprozesses von Entwickler1 vorgenommen. Für das Projekt, im Rahmen dessen die Software der DPP-Sitzungen entwickelt wurde, war er verantwortlich. Die Entwicklung war noch in der Spezifikationsphase, d.h. die

Spezifikationsdokumente waren noch vorläufig, und es hatte noch kein Review-Meeting darüber stattgefunden. Die Entwickler sollten in den DPP-Sitzungen ausnahmsweise ohne feste Anforderungen gemeinsam an der Entwicklung neuer Funktionalitäten für eine VoIP-Software bzw. an einem Prototyp als *Proof of Concept*[3] arbeiten.
Da Entwickler1 die Aufgabe hatte, die Feature-Server-Entwicklung in Bangalore anzusteuern, teilte er die Aufgaben zwischen sich und den beiden Entwicklern aus Bangalore auf.
Entwickler1 teilte jedem der Entwickler aus Bangalore eine eigene Aufgabe zu. Entwickler2 bekam die Aufgabe, ein Netzwerk-Feature (weiterhin mit SNMP-Feature bezeichnet) zu entwickeln. Entwickler3 bekam die Aufgabe, ein Abrechnungs-Feature (weiterhin als CDR-Feature bezeichnet) zu entwickeln. Nachdem die Entwickler ihre Einzelaufgaben beendet hatten, wurde als weitere Aufgabe, in den Sitzungen ein Aufzeichnungs-Feature (weiterhin als Voice Recording-Feature bezeichnet) zu erstellen, festgelegt.

Erzeugung durch	Aufgabe
Entwickler2	SNMP-Feature
Entwickler3	CDR-Feature
Entwickler1 und Entwickler2	Voice Recording-Feature

Tabelle 4.7 Aufteilung der Entwicklungsaufgaben zwischen den Entwicklern im DPP-Projekt

4.5 Sitzungstypen in den DPP-Sitzungen

Zu Beginn des DPP-Projektes waren die Sitzungen inhaltlich noch nicht so weit durchgeplant, dass schon feststand, welche Sitzungstypen in den Sitzungen zum Einsatz kommen würden. Entwickler1 war dafür verantwortlich, die Sitzungsinhalte festzulegen. Gespräche mit ihm während der Planungsphase ergaben, dass die Sitzungen folgende Inhalte haben könnten:

1. Durchsichten zur Defektbehandlung und zum Abgleich mit vorhandenen Spezifikationsdokumenten
2. Anlernen für Wissenstransfer zwischen den C5-Teams in Wien und Bangalore
3. Weiterentwicklung eines Prototyps
4. Übergabe der Spezifikation zur Entwicklung
5. Übergabe der Anwendung zur Wartung

Den tatsächlichen Inhalten der Sitzungen wurden später von mir die passenden Sitzungstypen entsprechend den in Abschnitt 2.3 erörterten Sitzungstypen zugeordnet. Insgesamt gab es drei verschiedene Sitzungstypen in den DPP-Sitzungen. Hiervon ausgenommen ist der Sitzungstyp Debugging aus der 4. Sitzung. Diese fiel wegen technischer Probleme aus. Der erste Sitzungstyp, Pilot, beschreibt Sitzungen, deren Inhalt sich auf die Einführung des DPP-Projektes beschränkte (siehe Abschnitt 4.7.1 über die Pilotphase). Des Weiteren gab es Sitzungen, in denen Quellcode erzeugt wurde (Pair-Developing) und andere, in denen Durchsichten durchgeführt wurden (Review). Dies entsprach auch den geplanten Aufgaben, die mir von Entwickler1 geschildert wurden (siehe Abschnitt 4.4.4).

[3] *Proof of Concept* entspricht hier einem Beweis für die mögliche Umsetzung der Spezifikation in lauffähigen Quellcode.

4.5.1 Review

Die Beschreibung des Sitzungstyps Review und ein Anwendungsfall wurden auf Basis der 7. Sitzung erstellt. Eine detaillierte chronologische Beschreibung sowie der Anwendungsfall (DPP-Walkthrough) befinden sich in Anhang F. Die beiden anderen als Review gekennzeichneten Sitzungen verliefen sehr ähnlich. An dieser Sitzung waren Entwickler1 und Entwickler3 beteiligt.
Entwicker3 hatte von Entwickler1 die Aufgabe bekommen, eigenständig ein Modul zu entwickeln (siehe Tabelle 4.7), für dessen Weiterentwicklung und Wartung er verantwortlich war. Als Ausgangspunkt dienten ihm vorläufige Anforderungs- und Spezifikationsdokumente, die Entwickler1 verwaltete. Im Fragebogen hatten beide Entwickler angegeben, dass es sich bei der zu erledigenden Aufgabe um ein Review handeln würde.
Wie mir Entwickler1 mitteilte, gehörten Reviews zwischen den Entwicklern nicht zum üblichen Standardentwicklungsprozess des C5-Teams (siehe Abschnitt 4.4.4). Über die genaue Umsetzung der Sitzungsziele hatten sie sich ebenfalls vorher nicht abgesprochen. Die daraus resultierende Problematik wird in Abschnitt 5.2.1 detaillierter erörtert. In der Sitzung hatte Entwickler1 folgendes Ziel formuliert: *„The goal is to get a short intro into the code"*. Entwickler3 hatte Folgendes im Fragebogen vor der Sitzung angegeben: *„Mostly knowledge and experience sharing, clarifying the approach before it's too late."* Entwickler1 wollte über die Präsentation den von Entwickler3 erzeugten Code kennenlernen, um einen Überblick über die Entwicklung zu behalten. Entwickler3 hoffte, durch die Erfahrungen von Entwickler1 Bestätigung für die korrekte Umsetzung der Anforderungen und weitere Hilfe an unklaren Stellen zu bekommen.

Der Ablauf der Sitzung kann folgendermaßen zusammengefasst werden. Die Sitzung dauerte ab dem Zeitpunkt, nach dem das Projekt beiden Entwicklern in synchronisierter Form vorlag, noch etwa 46 Minuten. Davon wurden insgesamt 24 Minuten mit Saros bzw. Eclipse gearbeitet. Entwickler1 war Host und hatte Entwickler3 zur Projektsynchronisation eingeladen. Entwickler3 bekam am Anfang der Sitzung von Entwickler1 die Driver-Rolle und blieb bis zum Ende Driver. Entwickler3 erklärte, wie sich der von ihm erzeugte Code verhielt. Da die vorläufigen Spezifikationsdokumente manche Punkte nicht abdeckten, ließ er sich an diesen Stellen das korrekte Verhalten seines Programmes von Entwickler1 bestätigen.
Für die Erklärungen öffnete Entwickler3 insgesamt 11 Dateien, markierte manche Stellen, über die er redete, mit dem Cursor, machte aber keine Änderungen im Code. Es wurden am Anfang und zwischendurch konkrete Entwurfsfragen von Entwickler3 gestellt. Teilweise wurde nach kurzen Diskussionen zwischen den Entwicklern von Entwickler1 Entwurfsentscheidungen getroffen, die er sofort in den Spezifikationsdokumenten notierte. In unklaren Fällen wurden die Entwurfsentscheidungen auf einen späteren Zeitpunkt verschoben.
Außer den Quellcode-Dateien in Saros wurden noch vorläufige Spezifikationsdokumente (FAD/TAD) betrachtet und von Entwickler1 angepasst sowie ein Testanruf mit der erzeugten Software von Entwickler3 durchgeführt.

Aus den Sitzungsinhalten ließen sich folgende Schlussfolgerungen bezogen auf den Sitzungstyp ziehen. Es handelte sich um eine bestimmte Art von Review, die in Kapitel 2 als Walkthrough[4] vorgestellt wurde. Das Sitzungsziel war, einen *Proof of Concept* zu diskutieren und anhand der Projekterfahrung von Entwickler1 zu beurteilen, ob die Implementierung auf eine sinnvolle Weise vorgenommen worden war. Dafür erklärte Entwickler3, wie er das erwartete Verhalten umgesetzt hatte und welche Punkte ihm aufgrund der unvollständigen Dokumente noch unklar waren. Entwickler1 entschied anhand seines Domänenwissens, wie mit unklaren Punkten, z.B. der optimalen Stelle einer Variableninitialisierung, umgegangen werden sollte. Er konzentrierte sich bei der Betrachtung des Codes nur auf semantische Korrektheit und ließ syntaktische Aspekte weitestgehend unberücksichtigt. Es handelte sich also um ein Review mit dem Ziel, den vorläufigen Entwurf und den vorhandenen Code auf semantischer Ebene zu verfeinern.

[4] Anm.: übliche Walkthroughs, wie sie in Abschnitt 2.3.2 von Wiegers definiert werden, setzen voraus, dass der Autor nicht selbst durch den Code führt. Dies wird auch in Abschnitt 5.2.1 diskutiert.

4.5.2 Pair-Developing

Die Beschreibung des Sitzungstyps Pair-Developing wurde anhand von Sitzung 5 erstellt. Weitere Sitzungen desselben Typs sind ähnlich verlaufen. Sie unterschieden sich nur durch verschiedene Subtypen wie z.B. Planung, Ad-hoc-Tests oder Problembehandlungen. An Sitzung 5 nahmen Entwickler1 und Entwickler2 teil. In Anhang E und F befinden sich eine detaillierte chronologische Beschreibung der Sitzung und ein Anwendungsfall zu Pair-Developing.

Es war die erste Sitzung des Typs Pair-Developing. Die geplanten Aufgaben für die Sitzung waren (1) existierenden Code von Entwickler1 anzuschauen und (2) daran gemeinsam weiterzuentwickeln.
Das geplante Sitzungsziel war, Entwickler2 über den Status der Entwicklung zu informieren, sodass die Entwickler in Zukunft gemeinsam daran weiterarbeiten konnten.

Der tatsächliche Ablauf der Sitzung kann folgendermaßen zusammengefasst werden. Die Sitzung teilte sich in drei Aktivitäten (Präsentation, Planung und Codieren) auf. Insgesamt dauerte die Sitzung nach der Synchronisation noch 44 Minuten. Die meiste Zeit hatten die Entwickler ihren Fokus auf die Editorfenster von Saros gerichtet.
Entwickler1 hatte zwei Monate vorher teilweise mit der Implementierung angefangen. Er war Host der Sitzung und stellte im ersten Teil der Sitzung (als Driver) seinen Code kurz vor, indem er die Struktur erklärte und wichtige Klassen öffnete, um darin einzelne Methoden und wichtige Parameter zu benennen. Danach zeigte er (mit dem Cursor und Markierungen) Entwickler2 zu implementierende Methoden, die durch eine „TODO“-Markierung gekennzeichnet waren. Am Ende der Erklärung zeigte Entwickler1 noch verschiedene Stellen im Code, die bereits implementiert waren, aber noch nicht funktionierten und überarbeitet werden mussten. Diese waren ebenfalls mit „TODO“ markiert.
Im zweiten Schritt besprachen die Entwickler die Aufgabenverteilung der offenen Punkte (offene Aufgaben, Fehlerbehandlung und Erstellung von Spezifikationsdokumenten).
Der letzte Teil der Sitzung bestand darin, dass die Entwickler an einer Stelle, die zur Überarbeitung mit einem TODO gekennzeichnet war, Veränderungen vornahmen und weitere Methoden dazu schrieben. Bei dieser Gelegenheit bekam Entwickler2 für wenige Minuten zweimal die Driver-Rolle von Entwickler1 zugeteilt.
Entwickler1 öffnete 6 verschiedene Dateien und war die meiste Zeit (42 von 44 Minuten) in der Driver-Rolle. Er markierte dabei mit seinem Cursor und scrollte durch die Dateien. Entwickler2 beteiligte sich an der Sitzung, indem er Rückfragen über den Ursprung von Parametern stellte, mit Entwickler1 über Entwurfsentscheidungen diskutierte und bei der Planung durch Auskünfte über den Status anderer Aufgaben informierte. Am Ende der Sitzung bekam er für zwei Minuten die Driver-Rolle, um zwei kleine Methoden zu erstellen.

Aus den Sitzungsinhalten ließen sich folgende Schlussfolgerungen bezogen auf den Sitzungstyp ziehen. Entwickler1 hatte diese Sitzung als Hand-over bezeichnet. Doch sein Plan war, den vorhandenen Code in den Sitzungen weiter zu bearbeiten. Daher handelte es sich nicht wirklich um eine Übergabe von Code, sondern um die Einführung einer weiteren Praktik von PP, dem geteilten Code (engl. *Collective Ownership*)[5]. Die Inhalte der Sitzung und der Sitzungsablauf bestätigten, dass es sich um eine Pair-Developing-Sitzung zwischen einem Experten und einem Novizen mit dem Ziel, Entwickler2 in das vorhandene Projekt einzuführen handelte. Die meiste Zeit hatte Entwickler1 die Driver-Rolle in den Sitzungen. Er sprach auch insgesamt viel mehr als Entwickler2. Für diese Sitzung könnte man den Sitzungstyp Pair-Developing in der Beschreibung noch um die Subtypen Einführung und Planung erweitern.

[5] Danach soll es jedem Entwickler in den PP-Sitzungen möglich sein, den Code zu verändern [Bec99].

4.6 Ablauf und Organisation von Sitzungstagen und Maßnahmen

Der Etablierungsprozess dauerte etwa 4 Monate und bestand aus 16 Sitzungen. Die Planung des Etablierungsprozesses wurde am Anfang zum Teil durch Entwickler1 und durch mich in Absprache mit dem Saros-Team übernommen (siehe Tabelle 4.8). Später wurden einige Komponenten des Etablierungsprozesses im Zuge der Prozessoptimierung angepasst (siehe Tabelle 4.9 und Tabelle 4.10). Die Sitzungsdetails (Anzahl wöchentlicher Sitzungen, Inhalte, Dauer, Beginn und Ende) wurden ursprünglich von Entwickler1 festgelegt und nach Beginn der Sitzungen mit den anderen Teilnehmern der Sitzung jeweils abgestimmt. Alle weiteren Maßnahmen des Etablierungsprozesses (Befragungen, Organisation der Reflexionen, Lösungen von technischen Problemen, etc.) wurden von mir organisiert.

Geplanter Ablauf an Sitzungstagen bis zur 5. Sitzung	Zeitraum
Ausgabe und Beantwortung Fragebogen US	Ein Tag vor der Sitzung bis Sitzungsbeginn
Sitzungstage mit Reflexionen vor der Sitzung	[Reflexion 10:15]
Geplanter Sitzungsbeginn	10:30
Geplante Sitzungsdauer	ca. 60 Minuten
Sitzungsende	Nach Ansage von Entwickler1 in der Sitzung
Ausgabe und Beantwortung Fragebogen MRS	Nach der Sitzung bis Ende des Tages erwünscht

Tabelle 4.8 Geplanter Ablauf an Sitzungstagen bis zur 5. Sitzung, inkl. Reflexion, Fragebogen Upcoming Session (US) und Fragebogen Most Recent Session (MRS)

Änderungszeitpunkt	Ablaufänderung
Ab der 5. Sitzung	Sitzungsdauer festgelegt auf 90 Minuten
Ab der 7. Sitzung	Visuelle Unterstützung in Reflexionen durch Saros
Ab der 8. Sitzung	Reflexionen an Sitzungstagen
Ab der 11. Sitzung	Fragebogen MRS wird am Ende der Sitzung ausgefüllt
Ab der 12. Sitzung	Nur noch eine Reflexion pro Woche nach der letzten Sitzung
Ab der 14. Sitzung	Fragebogen US und MRS gekürzt

Tabelle 4.9 Änderungen im Ablauf an Sitzungstagen während des Etablierungsprozesses

Geplanter Ablauf an Sitzungstagen ab der 12. Sitzung	Zeitraum
Ausgabe und Beantwortung Fragebogen US	Ein Tag vor der Sitzung bis Sitzungsbeginn
Geplanter Sitzungsbeginn	10:30
Geplante Sitzungsdauer	ca. 90 Minuten
Sitzungstage mit Reflexionen nach der Sitzung	[Reflexion 11:45]
Ausgabe und Beantwortung Fragebogen MRS	Am Ende der Sitzung
Geplantes Sitzungsende	12:00

Tabelle 4.10 Geplanter Ablauf an Sitzungstagen ab der 12. Sitzung, inkl. Reflexion, Fragebogen Upcoming Session (US) und Fragebogen Most Recent Session (MRS)

Die Anzahl der Sitzungstage variierte. Sie richtete sich nach der Aufgabenplanung von Entwickler1 und der Verfügbarkeit der beteiligten Entwickler. Meistens gab es eine DPP-Sitzung pro Woche und diese fand mittwochvormittags statt. In der Planungsphase hatte Entwickler1 erwähnt, dass er zwei bis drei Sitzungen für je zwei Stunden pro Woche plane. Doch stellte sich im Verlauf heraus, dass aus Zeitmangel der Entwickler höchstens zwei Sitzungen pro Woche geplant werden konnten und meistens nur eine davon tatsächlich stattfand. Gab es in den DPP-Sitzungen viel zu tun, wurden für die nächste Woche bis zu zwei Sitzungen geplant. Wenn gerade andere Projekte dringlicher bearbeitet werden mussten, wurden für die ganze Woche alle Sitzungen kurzfristig per Email abgesagt. Weitere Erörterungen zu ausgefallenen oder verschobenen Sitzungen werden in Abschnitt 5.2.2 behandelt.

Die Sitzungsinhalte wurden von Entwickler1 nach dem Stand der aktuellen Aufgaben und des C5-Projektes festgelegt. Er teilte sie etwa eine Stunde vor der Sitzung per Chat den anderen Entwicklern mit oder wählte manchmal während einer Sitzung spontan neue Aufgaben aus „TODO"-Markierungen im Quellcode aus. Weitere Einblicke hierzu bietet der Abschnitt über Sitzungstypen (siehe Abschnitt 4.5) und die Übersicht über die Sitzungen (siehe 4.7).

Die Sitzungszeit wurde aufgrund des Zeitunterschiedes von 3,5 Stunden zwischen Bangalore und Wien und weiteren Verpflichtungen von Entwickler1 auf 10:30 - 12:00 Uhr (CET – mitteleuropäische Zeit) bzw. 14 - 15:30 Uhr (SIT – Standard Indian Time) festgelegt. Diese Zeit lag genau zwischen den Mittagspausen der Entwickler. Die Entwickler aus Bangalore kamen gerade vom Essen, Entwickler1 ging nach den Sitzungen direkt zum Essen.
Für die ersten Sitzungen galten diese Zeiten noch nicht. Der Sitzungsbeginn wurde von Woche zu Woche festgelegt, und das Ende variierte je nach geplanter Aufgabe. Eine andere Sitzungszeit, z.B. 13 Uhr CET konnte nicht gewählt werden, da Entwickler1 am Nachmittag meist andere Meetings hatte. Die Sitzungsdauer wurde zu Beginn des DPP-Projektes von Entwickler1 nicht kommentiert. Er plante Aufgaben, die etwa eine Stunde dauerten und beendete die Sitzungen, wenn er es für richtig hielt. Ab der 5. Sitzung wurde die Sitzungszeit dann explizit auf 90 Minuten festgelegt, weil die Vorbereitungsphase zuvor den Beginn der produktiven Arbeitszeit zu sehr verzögert hatte (siehe Tabelle 4.9). Die Verzögerungen entstanden durch Verspätungen, fehlende Vorbereitung der Entwickler und technische Probleme.

Die an der Sitzung teilnehmenden Entwickler füllten für jede Sitzung zwei Fragebogen aus, die ich ihnen per Email zusandte. Wie schon im Abschnitt über die schriftliche Befragung erörtert, fragten sie Informationen zur aktuellen Sitzung ab (siehe Abschnitt 3.2.2). Der Fragebogen Upcoming Session (kurz US) wurde vor jeder Sitzung ausgefüllt. Er enthielt Fragen zu geplanten Aufgaben, Erwartungen und Befürchtungen der Entwickler. Der Fragebogen Most Recent Session (kurz MRS) wurde nach jeder Sitzung ausgefüllt und enthielt Fragen über tatsächlich erledigte Aufgaben, erfüllte und unerfüllte Erwartungen sowie Informationen über den Erfolg der Sitzung. Exemplare dieser Fragebogen befinden sich in Anhang B. Die Entwickler gaben an, dass sie etwa fünf Minuten für die Beantwortung der Fragebogen brauchten.
Obwohl die Entwickler die Fragebogen (MRS) zu Sitzungsende vorliegen hatten, dauerte es manchmal eine Woche, bis ich sie ausgefüllt zurückbekam. Zum einen machte es mir mehr Arbeit, weil ich sie mehrfach daran erinnern musste, und zusätzlich erschwerte es meine Vorbereitung auf die Reflexionen und die zeitnahe Erfassung der Ereignisse. Zum anderen waren die Angaben im Nachhinein weniger genau und lückenhaft, weil die Entwickler sich an die Geschehnisse aus der Sitzung nicht mehr erinnern konnten. Ich nutzte die Gelegenheit, als die Entwickler in der 11. Sitzung einmal schneller als geplant mit ihren Aufgaben fertig waren, und bat sie darum, ihre Fragebogen noch in der Sitzung auszufüllen. Da dies die oben erwähnten Probleme löste, etablierte ich dieses Vorgehen als weiteres Vorgehen im Sitzungsablauf (siehe Tabelle 4.9).
Zur 14. Sitzung wurden die Fragebogen angepasst und gekürzt (siehe Anhang B). Zum einen hatten sich die Entwickler darüber beschwert, dass die Fragebogen eine zu große Belastung wären. Dies sollte ihnen Zeit sparen. Zum anderen hatte sich im Verlauf des Etablierungsprozesses gezeigt, welche Fragen für die weitere Auswertung besonders relevant waren und welche man evtl. weglassen konnte. Auswertungen und Ergebnisse der Fragebogen befinden sich in Abschnitt 5.3.2.

Wie in Abschnitt 3.2 bereits erläutert, sollten regelmäßige Reflexionen den Beteiligten während des Etablierungsprozesses die Möglichkeit des Informationsaustausches außerhalb der Sitzungen bieten. Bis zur 11. Sitzung gab es zu jeder Sitzung eine Reflexion (siehe Tabelle 4.8). Dazwischen lag mindestens ein Tag. An Sitzungstagen fanden die Reflexionen vor der nächsten Sitzung um 10:15 Uhr statt. Die Reflexionen dauerten meistens 10 Minuten und in Ausnahmefällen, z.B. zur Optimierung der Audioeinstellungen auch länger (siehe Tabelle 4.11).
Ab der 7. Sitzung wurde Saros auch zur visuellen Unterstützung der Reflexionen eingesetzt. Die zu besprechenden Themen wurden in eine Datei geschrieben und zwischen den Beteiligten in Saros geteilt.

Es wurde durch verschiedene Änderungen in Bezug auf die Reflexionen versucht, den Zeitaufwand gering zu halten und die Planung zu vereinfachen. Ab der 8. Sitzung wurden Reflexionen nicht mehr als Einzeltermine geplant, sondern an Sitzungstagen vor die nächste Sitzung gelegt (siehe Tabelle 4.9).
Der nächste Schritt war, Verspätungen zu Sitzungen, die durch Reflexionen entstanden, zu verhindern, z.B. durch Verschiebung des Termins für die Reflexion auf einen früheren Zeitpunkt. Schließlich wurde die Reflexion an das Ende der Woche auf den letzten Sitzungstermin verschoben da es bei zwei Sitzungen pro Woche zu viel Aufwand bedeutete, mehrere Reflexionen durchzuführen und sich herausstellte, dass Reflexionen vor der Sitzung meistens zu Verspätungen führten (weitere Erläuterungen auch im Abschnitt 5.2.3). Diese letzte Maßnahme erleichterte die Organisation der Reflexionen sehr. Die letzte Änderung wies eine kleine Schwachstelle auf, nachdem einmal die letzte Sitzung der Woche spontan abgesagt wurde. Denn auch die Reflexionen für alle Sitzungen dieser Woche fielen damit aus.

4.7 Überblick und Beschreibung der Phasen des Etablierungsprozesses

Zur Strukturierung des Forschungsprozesses wurde der Etablierungsprozess entsprechend der Forschungsmethode CAR (siehe Abschnitt 3.1) in Zyklen aufgeteilt.
Die Pilotphase galt als Vorbereitungsphase. Darauf folgten vier Arbeitszyklen (Tabelle 4.11). Diese Aufteilung diente der regelmäßigen Evaluierung des Etablierungsprozesses. Jeder Arbeitszyklus sollte unabhängig von inhaltlichen Ereignissen des Entwicklungsprozesses etwa vier Sitzungen umfassen. Die Aufteilung sollte die Organisation der Datenauswertung und den Umgang mit Erkenntnissen und Maßnahmen des Forschungsprozesses strukturieren.
Abbildung 4.3 zeigt die Teilschritte innerhalb eines Arbeitszyklus.
Jeder Arbeitszyklus begann mit einer Ist-Analyse über den Zustand des Etablierungsprozesses. Dafür wurden Ereignisse und Probleme anhand der Daten aus den Sitzungen, Reflexionen und Fragebogen zusammengefasst und ausgewertet. Auf Basis dessen wurden weitere Erkenntnisse über den Etablierungsprozess gesammelt.
Im zweiten Schritt des Arbeitszyklus, der Soll-Analyse, wurden anhand der Ergebnisse aus der Ist-Analyse neue Ziele für den Etablierungsprozess festgelegt. Für Absprachen über das strategische Vorgehen und Verfeinerungen von geplanten Maßnahmen fand zum Ende des zweiten Schrittes ein Meeting mit den Leitern des Saros-Teams statt. Darin wurden Ereignisse und Erkenntnisse entsprechend der Ist-Analyse und die beste Wahl weiterer Maßnahmen entsprechend der Soll-Analyse evaluiert.
Im dritten Schritt wurde dann in den Etablierungsprozess durch Interventionen, die sich aus den vorherigen Schritten ergeben hatten, eingegriffen. Interventionen, die konfliktbehaftet waren wie z.B. der Umgang mit unerfüllten Erwartungen, wurden eher in mündlichen Gesprächen im Rahmen der Reflexion oder in Einzelgesprächen besprochen. Einfache Interventionen wie z.B. Einstellungsänderungen an der technischen Infrastruktur, wurden auch in schriftlicher Form als Empfehlungen per Email kommuniziert. Auf die Interventionen folgte im nächsten Schritt wieder eine Ist-Analyse, um zu erfassen, wie sich der Etablierungsprozess inklusive der Interventionen seitdem entwickelt hatte. Dies leitete wieder einen neuen Arbeitszyklus ein.

Diese Teilschritte galten als Orientierung für den Forschungsprozess. Der tatsächliche Etablierungsprozess erforderte oft auch Analysen und Interventionen innerhalb eines Arbeitszyklus. Bei akuten Problemen, z.B. mit der technischen Infrastruktur, wurde natürlich nicht das nächste Meeting des Arbeitszyklus abgewartet, sondern sofort Maßnahmen zur Lösung ergriffen.
Die Arbeitszyklen boten unabhängig von täglichen Anforderungen des Etablierungsprozesses eine zusätzliche regelmäßig stattfindende Prüfung des aktuellen Status und der geeignetsten Vorgehensweise.

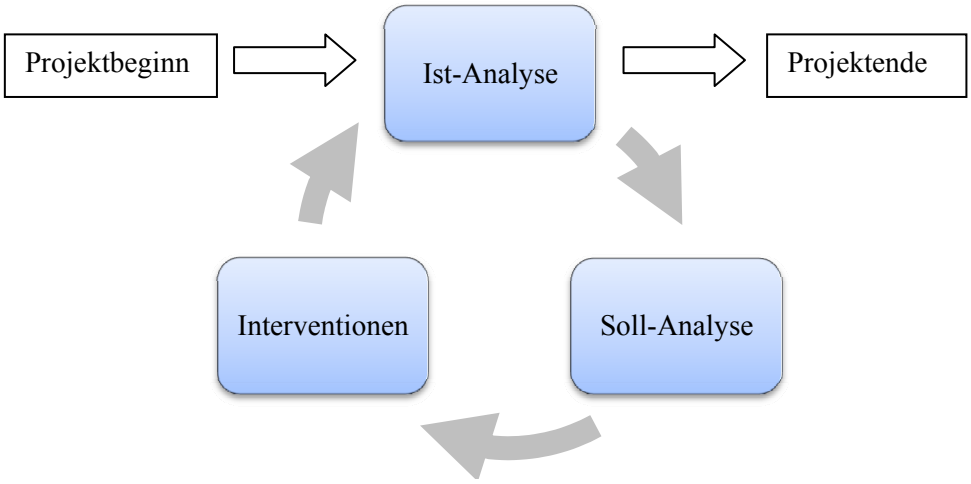


Abbildung 4.3 Aufteilung und Ablauf der Arbeitszyklen des Forschungsprozesses innerhalb des Etablierungsprozesses

Wie Tabelle 4.11 zeigt, bestand der Etablierungsprozess aus 16 Sitzungen. In Abschnitt 4.5 wurden die darin aufgeführten Sitzungstypen bereits beschrieben. Insgesamt gab es zwei Pilotsitzungen, drei Reviews und 11 Pair-Developing-Sitzungen. An den Reviews war sowohl Entwickler2 als später auch Entwickler3 beteiligt. Die Pair-Developing-Sitzungen wurden hauptsächlich mit Entwickler2 durchgeführt. Entwickler1 nahm an allen Sitzungen teil. Jeder Sitzungsverlauf (außer den Pilotsitzungen) wurde mit den Entwicklern in einer gemeinsamen Reflexion erörtert.

In der Pilotphase (1-2) wurde den Entwicklern die technische Infrastruktur der DPP-Sitzungen (besonders Saros) vorgestellt und die weitere Planung des Etablierungsprozesses besprochen.

Im ersten Arbeitszyklus (3-7), der sechs Wochen dauerte, fanden erste produktive Sitzungen (d.h. Sitzungen, in denen die Entwickler ihrer Arbeit nachgehen konnten) statt. Es wurden zudem Sitzungsinhalte festgelegt, verschiedene Sitzungstypen ausprobiert, erste technische Probleme behandelt und regelmäßige Sitzungen etabliert.

Im zweiten Arbeitszyklus (8-11), der weitere 6 Wochen dauerte, wurden erhebliche technische Probleme gelöst, der Etablierungsprozess um Sitzungen mit drei Entwicklern und dem Multi-Driver-Mode erweitert und erste Konflikte in der Zusammenarbeit der Entwickler aufgedeckt.

Der dritte Arbeitszyklus (12-13), enthielt wegen Sitzungsausfällen nur zwei Sitzungen. Er dauerte vier Wochen. Währenddessen wurde den Entwicklern mehr Verantwortung im Etablierungsprozess übergeben, und es fanden erste tiefere Analysen der Ereignisse im Entwicklungs- und Etablierungsprozess statt. Durch das Arbeiten mit fremdem Code änderten sich in diesem Arbeitszyklus ebenfalls die Rollenverteilung und die Zusammenarbeit der Entwickler.

Der vierte und letzte Arbeitszyklus (14-16) ging über einen Zeitraum von 3 Wochen. Er bestand aus drei Sitzungen.

Eine Beschreibung des Entwicklungs- und Etablierungsprozesses sowie der Interventionen folgen nun im Detail. Die Beschreibung der Sitzungen ist aus Gründen der Übersichtlichkeit nicht vollständig, sondern beschränkt sich auf wesentliche Vorkommnisse aus den Sitzungen.

Sit-zung	Projekt-phase	Datum	Sitzungstyp	Beteiligte Entwickler	Dauer in h[6]	Reflexion am[7]	Dauer in h
1.	Pilotpha-se	2009-04-16	Pilot	E1	00:41		
2.		2009-04-22	Pilot	E1,E2	00:36		
3.	1. Zyklus	2009-04-29	Review	E1,E2	01:16	2009-04-30	00:09
4.		2009-05-20	Debugging	E1,E2	01:08	ausgefallen	
5.		2009-05-27	Pair-Developing	E1,E2	00:59	2009-05-28	00:08
6.		2009-06-03	Pair-Developing	E1,E2	01:34	2009-06-04	00:12
7.		2009-06-10	Review	E1,E3	01:16	2009-06-17	00:42
8.	2. Zyklus	2009-06-17	Pair-Developing	E1,E2	01:06	2009-06-23	00:28
9.		2009-06-23	Review	E1,E2,E3	00:48	2009-06-24	00:25
10.		2009-06-24	Pair-Developing	E1,E2,E3	01:25	2009-06-25	00:36
11.		2009-07-01	Pair-Developing	E1,E2	01:06	2009-07-08	00:10
12.	3. Zyklus	2009-07-08	Pair-Developing	E1,E2	01:08	ausgefallen	
13.		2009-07-15	Pair-Developing	E1,E2	00:56	2009-07-15	00:10
14.	4. Zyklus	2009-08-05	Pair-Developing	E1,E2	00:59	2009-08-05	00:22
15.		2009-08-11	Pair-Developing	E1,E2	01:33	2009-08-12	siehe Nr.16
16.		2009-08-12	Pair-Developing	E1,E2	01:16	2009-08-12	00:15

Tabelle 4.11 Übersicht über Arbeitszyklen, DPP-Sitzungen und Reflexionen im DPP-Projekt

4.7.1 Pilotphase

4.7.1.1 Planung und Ziele der Pilotphase

Die Pilotphase diente der Vorbereitung auf die DPP-Sitzungen. In dieser Phase sollten sich alle Beteiligten der DPP-Sitzungen zusammenfinden, um die technische Infrastruktur kennenzulernen und gemeinsam über Ziele und Ablauf des Etablierungsprozesses im Sinne des Forscher-Teilnehmer-Abkommens (siehe RCA im Abschnitt 3.1) zu sprechen.
Bisher hatte ich nur persönlichen Kontakt zu einem der drei beteiligten Entwickler (Entwickler1) gehabt. Die Entwickler aus Bangalore wurden bis zu diesem Zeitpunkt per Kopien (per *Carbon Copy, kurz* CC) des Emailverkehrs über die Planung auf dem Laufenden gehalten.
In der Pilotphase war eine Sitzung geplant, in der die beteiligten Entwickler die Saros-Funktionalitäten kennenlernen sollten. Die zweite Sitzung sollte bereits teilweise für den produktiven Einsatz von DPP genutzt werden. Daraus ergaben sich folgende Maßnahmen für die Pilotphase:

1.	Einführung der Entwickler in Saros und die technische Infrastruktur
2.	Erster Einsatz von DPP im Entwicklungsprozess
3.	Weitere Planung und Besprechung der Ziele für den Etablierungsprozess

Tabelle 4.12 Maßnahmen in der Pilotphase

[6] Entspricht der Dauer der Aufzeichnung in Stunden
[7] Bis zur 11. Sitzung fanden Reflexionen an Sitzungstagen vor der Sitzung statt, ab der 12. Sitzung danach

4.7.1.2 Chronologische Beschreibung der Ereignisse in den Sitzungen

Sitzung 1:
Die erste Pilotsitzung diente als Einführung von Entwickler1 in die technische Infrastruktur und die Funktionalitäten der von der AG SE bereitgestellten Software für DPP-Sitzungen (Saros). Dabei wurden Benutzerkonten eingerichtet und wichtige Funktionalitäten, die eine selbstständige Anwendung von Saros ermöglichen, von mir angesprochen. Es wurden dabei folgende Funktionalitäten näher erläutert: Verbindung zum Server herstellen, das Projekt synchronisieren, Rollenwechsel, Auswirkungen durch Änderungen des Drivers und die Awareness-Funktionalitäten (inklusive Verfolgermodus).[8]
Außerdem gab es weitere Absprachen über geplante Sitzungen und den Etablierungsprozess. Der geplante Sitzungsablauf mit allen weiteren Maßnahmen wurde, wie in Tabelle 4.8 im Abschnitt über den Sitzungsablauf dargestellt, von mir erläutert.

Sitzung 2:
In der zweiten Sitzung waren zwei Aufgaben geplant. Erstens war eine weitere technische und Prozesseinführung für die Entwickler aus Bangalore geplant. Zweitens sollte es der erste produktive Einsatz von DPP werden. Alles kam anders als geplant. Entwickler3 erschien nicht zur Sitzung, und einen Tag vor Beginn der Sitzung teilte mir Entwickler1 mit, dass die erste produktive Sitzung um eine Woche verschoben werden müsste, weil mir die endgültige Zusage von Teles fehlte, um deren Arbeit einzusehen.
Die Entwickler konnten in der Sitzung zwar nicht wie geplant an ihrem Projekt arbeiten, doch für die geplante Prozesseinführung (wie in der ersten Sitzung) brauchte ich keine Freigabe. Daher konnte die Sitzung wenigstens genutzt werden, um auch Entwickler2 in den Etablierungsprozess einzuführen.

4.7.1.3 Besondere Ereignisse und Probleme

Eigentlich sollte nur eine Pilotsitzung stattfinden. Bis zur zweiten Sitzung hatte ich jedoch noch keine Freigabe, die Inhalte der Arbeit bei Teles einzusehen. Daher verzögerte sich die praktische Umsetzung des Projektes auf die dritte Sitzung, und es wurde nicht, wie geplant, nur eine Pilotsitzung für Entwickler1, sondern noch eine weitere für Entwickler2 durchgeführt. Entwickler3 erschien nicht, wie erwartet, zur 2. Sitzung.

Probleme mit der technischen Infrastruktur (besonders mit Saros) unterbrachen den Ablauf der Sitzungen mehrmals. Weder die verwendete technische Infrastruktur noch Saros waren bis zur Pilotphase in einem Zustand, der eine störungsfreie Sitzung möglich gemacht hätte. Saros befand sich zu diesem Zeitpunkt noch in der α-Version der Entwicklungsphase. Daher kam es bereits in der Pilotphase, die mit einem kleinen Beispielprojekt durchgeführt wurde, zu mehrfachen Unterbrechungen des Ablaufs aufgrund von technischen Problemen.
Es gab drei gravierende technische Probleme. Erstens hatte Entwickler2 in den ersten Minuten Probleme, sich über die verwendete Audiosoftware (Mumble) mit uns in Verbindung zu setzen. Zweitens gab es Schwierigkeiten mit dem Verbindungsaufbau von Saros im Netzwerk. Es wurde von Saros zur Übertragung zwischen Entwickler1 (dem Host) und mir nur eine langsamere Verbindung über den Server statt eine direkte P2P-Verbindung[9] aufgebaut. Drittens funktionierten grundlegende Saros-Funktionalitäten wie der Rollenwechsel, der Verfolgermodus und die Behandlung von Inkonsistenzen, nicht wie erwartet. Beispielsweise konnte Entwickler1 nach mehreren Rollenwechseln den Verfolgermodus als Observer nicht mehr deaktivieren.

[8] Weitere Erläuterungen zu Saros befinden sich in Kapitel 2.5
[9] Peer-To-Peer (kurz P2P): eine Direktverbindung zwischen Rechner ohne Vermittlung durch einen Server.

4.7.1.4 Ergebnisse der Pilotphase

Die Ziele der Pilotphase konnten bis auf die Prozesseinführung von Entwickler3 und den ersten produktiven Einsatz von DPP umgesetzt werden. Es ergaben sich schwerwiegende Probleme mit der technischen Infrastruktur, die beseitigt werden mussten.
Ich bekam die Gelegenheit, Entwickler2 aus Bangalore kennenzulernen, und mich und meine Rolle im DPP-Projekt auch bei ihm vorzustellen. Gleichzeitig hatte ich die Möglichkeit, die für die Aufzeichnung der Sitzungen aufgesetzte Infrastruktur im praktischen Einsatz zu testen.[10]

4.7.1.5 Sich ergebende Maßnahmen für die nächste Phase

Die technischen Probleme, die sich in den Pilotsitzungen (und auch später) gezeigt hatten, mussten analysiert, beschrieben und an die Saros-Entwickler weitergegeben werden (siehe Abschnitt 5.3.6 für weitere Informationen über den Einfluss der DPP-Sitzungen auf die Saros-Entwicklung).
Vor Beginn der produktiven Sitzungen wurden letzte Verbesserungen an den Fragebogen und den Leitfäden für die Sitzungen und die Reflexionen vorgenommen. Exemplare der Fragebogen und Leitfäden befinden sich in Anhang C.

4.7.2 Erster Arbeitszyklus

4.7.2.1 Ziele und Maßnahmen für den ersten Arbeitszyklus

Ziel des ersten Arbeitszyklus war es, DPP-Sitzungen bei Teles zu etablieren und erste Erkenntnisse über den Nutzen des Etablierungsprozesses zu gewinnen. Dabei sollten soweit wie möglich technische Probleme behandelt und ausgeräumt werden.
Zu den konkreten Maßnahmen für die Sitzungen gehörte sicherzustellen, dass mindestens eine Sitzung in der Woche mit den Teles-Entwicklern stattfand. Außerdem sollte durch Beobachtung und Befragung sichergestellt werden, dass die inhaltlichen Ziele der Sitzungen erreicht werden konnten.
Weiterhin sollten erste Erkenntnisse über die vorgekommenen Sitzungstypen und –verläufe sowie der Ablauf und Nutzen von Reflexionen analysiert werden. Daraus ergab sich folgende Liste von Maßnahmen:

1.	Inhaltliche Ziele, Erwartungen und Erfahrungen der Entwickler ermitteln
2.	Prüfen, ob inhaltliche Ziele der Entwickler erreicht wurden
3.	Eine DPP-Sitzung pro Woche etablieren
4.	Sitzungstypen und -verläufe analysieren
5.	Technische Probleme behandeln oder beseitigen
6.	Ablauf und Nutzen der Reflexion bewerten

Tabelle 4.13 Maßnahmen im ersten Arbeitszyklus

4.7.2.2 Chronologische Beschreibung der Ereignisse in den Sitzungen

Sitzung 3:
Die erste produktive Sitzung war ein Review[11], indem Entwickler2 den Code, den er vorher alleine erzeugt hatte (SNMP-Feature), präsentierte, und die Entwickler sich im Zuge dessen über Ent-

[10] Weitere Details zum Aufzeichnungssetting befinden sich in Abschnitt 3.3.
[11] Details über Reviews sind in Abschnitt 4.5.1 erläutert.

wurfsentscheidungen austauschten. Nach der Vorbereitungsphase (inkl. Projektsynchronisation) blieben noch 36 Minuten produktive Arbeitszeit[12].
Einer der Gründe für die lange Vorbereitungsphase war die Größe des Projektes, das synchronisiert werden sollte. Das Projekt hatte nun nicht mehr eine Größe unter 1MB, wie das Beispielprojekt in der Pilotphase, sondern es sollte ein Projekt geteilt werden, das ca. 100 MB groß war. Plötzlich spielte die Netzwerkverbindung (ob per IBB über den Server oder direkt per P2P) eine noch größere Rolle als bisher, da hiervon insbesondere abhing, wie schnell die Synchronisation des Projektes möglich war.
Auch die unzureichende Audioqualität in Form von Echos, metallischen Geräuschen in der Leitung, lauten Hintergrundgeräuschen und Verzerrungen der Stimmen störte den Ablauf der Sitzung.
Die grundlegendsten Saros-Funktionalitäten wie z.B. Einladung zum Projekt (Share Project), Rollenwechsel, Verfolgermodus und Konsistenzwiederherstellung ließen sich nicht störungsfrei anwenden.

Sitzung 4:
Seit der letzten Sitzung waren wegen Ausfällen durch Krankheit meinerseits und dringlicherer Aufgaben von Entwickler1 drei Wochen vergangen. Für die 4. Sitzung war spontan die Behebung von zwei *Exceptions* (als Sitzungstyp Debugging gekennzeichnet) geplant.
Entwickler1 äußerte sich im Fragebogen vor der Sitzung erneut über seine Einstellung zu geeigneten Sitzungsinhalten für DPP-Sitzungen: *"task [Determine the cause for 2 exceptions . Fix the problems] is not really suitable. The reason for the task is the importance"*.
Zu den technischen Problemen aus den vorherigen Sitzungen kamen zwei weitere Probleme hinzu. Zum einen hatten die Entwickler vorher nicht viel mit Saros gearbeitet und waren daher in der Anwendung noch ungeübt. Zum anderen bereiteten sie sich nur sehr kurzfristig auf die Sitzungen vor. So wurde die 4. Sitzung abgebrochen, weil Entwickler1 das vorher von mir empfohlene Saros Update auf das neueste Release erst in der Sitzung machte, nachdem Kompatibilitätsprobleme seiner Version mit der neueren Version von Entwickler2 auftraten.

Sitzung 5:
Entwickler2 verspätete sich etwa 17 Minuten zur ersten Sitzung vom Typ Pair-Developing.
Nach der Vorbereitungsphase blieben den Entwicklern weitere 44 Minuten für die produktive Sitzung. Entwickler1 stellte den von ihm vorher erzeugten Code (Voice Recording-Feature) kurz vor, damit sie gemeinsam daran weiterarbeiten konnten (siehe auch die Beschreibung in Abschnitt 4.5.2).
In dieser Sitzung konnten nun endlich letzte technische Probleme bezüglich der direkten Netzverbindung (P2P) zwischen allen Beteiligten bereinigt werden. So konnte auch ich mittlerweile mit den Entwicklern das Projekt über Saros teilen. Da ich inhaltlich nicht an dem Projekt der Entwickler beteiligt war, konnte ich Arbeitsschritte, die außerhalb der Editorfenster von Eclipse gestartet wurden, z.B. Spezifikationsdokumente sowie Testläufe und -berichte auf dem Server, nicht einsehen. Daher richtete Entwickler1, der hauptsächlich die Tests durchführte, mit mir eine Share-Desktop-Verbindung ein, die es mir ermöglichte, alle seine Aktionen im Workspace zu sehen.
Entwickler1 hatte die meiste Zeit die Driver-Rolle. In den wenigen Minuten, die Entwickler2 die Driver-Rolle hatte, traten Inkonsistenzen auf. Da er nicht kommentiert hatte, was er tat, bemerkten wir die Inkonsistenzen erst, als er mit der Implementierung fertig war, sodass der Code noch einmal geschrieben werden musste (siehe Abschnitt 5.2.4 für detaillierte Erörterungen).
Am Ende der Sitzung herrschte zur Abwechslung einmal ausgelassene Stimmung. Alle Beteiligten waren zufrieden und motiviert. Die Freude über die funktionierende technische Infrastruktur war spürbar. Dies zeigte sich durch gemeinsames Lachen und dadurch, dass Entwickler1 in der Reflexion zu dieser Sitzung von selbst Möglichkeiten überlegte, wie man den Etablierungsprozess noch weiter ausbauen könnte. Daher schlug er vor, an der nächsten Sitzung noch den dritten Entwickler zu beteiligen.

[12] Abbildung 5.4 enthält eine Übersicht über die Dauer der Vorbereitungsphase im Vergleich zur produktiven Arbeitszeit aller Sitzungen.

Sitzung 6:
Inhaltlich setzte Sitzung 6 das Pair-Developing am Voice Recording-Feature fort. Entwickler3 nahm nicht wie geplant an der Sitzung teil. Nach dem Ablauf der 5. Sitzung war ich guter Hoffnung, dass die anfänglichen technischen Probleme endlich überwunden waren und nun der Fokus nur noch auf die inhaltliche Gestaltung der Sitzung übergehen könnte. Obwohl sich die Entwickler mit Vorbereitungen für die Sitzung sonst eher zurückhielten (vor allem, wenn es darum ging, Fragebogen auszufüllen oder wie in Sitzung 4 ein Update durchzuführen), hatten sie inzwischen ihre Software eigenständig auf die neueste verfügbare Version von Saros aktualisiert. Dies hatte zwei Folgen: erstens, dass ich die Sitzung nur über die Netviewer-Sitzung verfolgte, weil mein Saros-Update nicht schnell genug beendet war, und zweitens, dass nach dem Update wieder neue technische Probleme auftraten. Beispielsweise funktionierte der Verfolgermodus nach dem ersten Rollenwechsel nicht mehr.
In der Sitzung hatte sich Entwickler1 nichts anmerken lassen, doch im Fragebogen nach der Sitzung stellte sich heraus, dass er bei der Untersuchung für die Gründe von verschiedenen Problemen (Exceptions) lieber alleine gearbeitet hätte, weil er seiner Meinung nach alleine schneller gewesen wäre.

Sitzung 7:
Die 7. Sitzung war die zweite Review-Sitzung und die erste Sitzung mit Entwickler3. Die Entwickler klärten kurz, wie sie vorgehen wollten. Dann stellte Entwickler3 den von ihm entwickelten Code (CDR-Feature) vor, und die Entwickler diskutierten über Entwurfsentscheidungen anhand der vorläufigen Spezifikationsdokumente (siehe auch die Beschreibung des Ablaufs in Abschnitt 4.5.1).
Bis zu Sitzungsbeginn war ich davon ausgegangen, dass dies die erste Sitzung mit drei aktiven Teilnehmern sein würde. Doch Entwickler2 schaltete sich nach einigen Minuten ein, um mitzuteilen, dass er nicht an der Sitzung teilnahm.
Endlich lernte ich auch den dritten Entwickler des Projektes kennen. Wir hatten bisher nur per Email Kontakt gehabt und sprachen nun das erste Mal über die Audioverbindung miteinander. Ich hatte ihm zur Vorbereitung alle relevanten Informationen über den Etablierungsprozess und die technische Infrastruktur schriftlich mitgeteilt. Mein Angebot, mit ihm auch eine Demonstration von Saros durchzuführen, ignorierte er.
Bisher hatte ich angenommen, die Entfernung nach Indien müsste dafür verantwortlich sein, dass der dortige Entwickler leise und manchmal kaum hörbar war. Doch Entwickler3 belebte die Sitzung allein durch seine laute und volle Stimme, die keinen Zweifel über seine Präsenz zuließ.
Leider konnte dies nicht die Tatsache kompensieren, dass er wegen mangelnder Erfahrung im Umgang mit Saros den Sitzungsbeginn erheblich aufhielt. Er wählte bspw. nicht die Saros-Option, ein bereits existierendes Projekt zu teilen, was die Synchronisationszeit um mehrere Minuten verlängerte. Dies führte auch dazu, dass ein weiterer Synchronisationsversuch scheiterte, sodass erst nach etwa 30 Minuten mit der produktiven Sitzung begonnen werden konnte. Entwickler3 kompensierte die lange Vorbereitungsphase ein wenig, indem er während der Synchronisation bereits das Gespräch auf Inhalte und Ziele der Sitzung lenkte.

4.7.2.3 Besondere Ereignisse und Probleme

Die lange Vorbereitungsphase in den Sitzungen und technische Probleme stellten ein wesentliches Problem im ersten Arbeitszyklus dar. Es vergingen teilweise bis zu 30 Minuten bis der Code (als Projekt in Eclipse) zwischen den Entwicklern geteilt und in synchronisierter Form vorlag (siehe Abbildung 5.4). Mit verschiedenen Maßnahmen wurde nach und nach versucht, dem entgegenzusteuern. Zu den Maßnahmen gehörte die explizite Verlängerung der Sitzungszeit auf 90 Minuten, was durch die unterschiedlichen Zeitzonen erschwert wurde. Zweitens wurde die Synchronisationszeit durch vorheriges Teilen des Projektes über das Projektarchiv bzw. die Versionsverwaltung verkürzt. Drittens erfolgten weitere Vorschläge für alternative Vorgehensweisen. Immer wenn es

nötig war, wurden in der Sitzung weitere Hinweise und Erklärungen über den effektiven Umgang mit Saros angesprochen.
Die 4. Sitzung musste wegen des späten Updates von Entwickler1 und weiteren technischen Problemen abgebrochen werden.
Entwickler1 erweiterte den Etablierungsprozess, indem er den Vorschlag machte, den dritten Entwickler in die Sitzung zu integrieren.

4.7.2.4 Ergebnisse des ersten Arbeitszyklus

Zu Beginn des ersten Arbeitszyklus bekamen die Entwickler einmalig einen allgemeinen Fragebogen (General Questions), in dem die Erwartungen an die Sitzungen und vorhandene Erfahrungen von DPP erhoben wurden. Die Ergebnisse daraus befinden sich in Abschnitt 4.4.1.
Über die zwei Fragebogen zu jeder Sitzung (US und MRS, siehe Abschnitt 4.6) und Beobachtungen in den Sitzungen, konnte überprüft werden, ob geplante Sitzungsziele (bis auf Sitzung 4) auch erreicht wurden. Weitere Ergebnisse aus den Fragebogen werden in Abschnitt 5.3.2 präsentiert.

Im ersten Arbeitszyklus konnten regelmäßige DPP-Sitzungen mit zwei Entwicklern etabliert werden. Der Verlauf der Sitzungen war jedoch, besonders am Anfang und am Ende, wenig strukturiert. Dies zeigte sich bspw. durch die lange Vorbereitungszeit zu Sitzungsanfang und das plötzliche Sitzungsende, dass durch Entwickler1 spontan festgelegt wurde.
Es kamen im ersten Arbeitszyklus sowohl Sitzungen vom Typ Review als auch vom Typ Pair-Developing vor. Die geplante Aufgabe von Sitzung 4, einen weiteren Sitzungstyp Debugging in den Sitzungen zu verwenden, löste die Notwendigkeit aus zu prüfen, inwiefern der Einsatz des Eclipse-Debuggers von Saros überhaupt unterstützt werden könnte und sollte. Es war vorher schon vom Saros-Team kommuniziert worden, dass Saros die Verwendung des Debuggers bzw. das Verwenden von ausführbaren Anwendungen nicht unterstützte, und dies war auch das Resultat einer erneuten Prüfung.
Die Sitzungen im ersten Arbeitszyklus waren vor allem durch neue technische Probleme, z.B. ein fehlgeschlagenes Update von Saros, und bereits aus der Pilotphase bekannte Probleme geprägt. Einige dieser Probleme, z.B. bzgl. der Synchronisationszeit, konnten im Verlauf des Arbeitszyklus gelöst werden.
Die Reflexionen im ersten Arbeitszyklus ermöglichten mir Rückfragen über den Hergang von Vorkommnissen aus den Sitzungen und Angaben der Entwickler in den Fragebogen. Beispielsweise wurde über Ursachen und Hergang von technischen Problemen aus den Sitzungen gesprochen. Außerdem wurden darin weitere Sitzungsdetails wie z.B. nächste Termine besprochen. Die Entwickler nutzten die Reflexionen jedoch nicht wie erwartet für den direkten Austausch von Erfahrungen aus dem Etablierungsprozess (siehe auch Abschnitt 5.3.3).

4.7.3 Zweiter Arbeitszyklus

4.7.3.1 Ziele und Maßnahmen für den zweiten Arbeitszyklus

Ziele des zweiten Arbeitszyklus waren, die Sitzungen bei Teles soweit wie möglich von technischen Problemen zu befreien, den Nutzen für die Entwickler zu erhöhen sowie den Fokus auf inhaltliche Aspekte zu lenken (Sitzungstyp, -ablauf, -organisation, etc.). Organisatorische Maßnahmen sollten die Kommunikation bzw. den Informationsfluss zwischen den Beteiligten verbessern und einen strukturierteren Arbeitsablauf herbeiführen. Zur Optimierung der Sitzungsstruktur war es nötig, den Standardentwicklungsprozess des C5-Teams kennenzulernen und den Ablauf anhand dessen zielorientierter zu gestalten. Verschiedene Mittel (Emails, Einzelgespräche und Gruppengespräche) sollten eingesetzt werden, um die Änderungsvorschläge an die Entwickler zu vermitteln. Technische Probleme, vor allem mit der Audiosoftware, sollten die Sitzung am Ende des Arbeitszyklus nicht weiter stören.

1.	Ursprünglichen Entwicklungsprozess genauer erforschen
2.	Strukturierten und zielorientierten Sitzungsablauf etablieren
3.	Technische Probleme (bzgl. Saros und der Audioverbindung) beheben
4.	Kommunikation zwischen den Entwicklern in der Reflexion etablieren
5.	Ablauf und Nutzen der Fragebogen und der Reflexion bewerten

Tabelle 4.14 Maßnahmen im zweiten Arbeitszyklus

4.7.3.2 Chronologische Beschreibung der Ereignisse in den Sitzungen

Sitzung 8:
In der Reflexion vor der Sitzung wurden unter anderem die Audio-Einstellungen mit allen Entwicklern durchgegangen und letzte Probleme damit beseitigt. Die Reflexion dauerte damit länger als erwartet, sodass die Sitzung erst mit einer Verspätung von 15 Minuten starten konnte. Während der 10 Minuten der Synchronisation und auch einmal innerhalb der Sitzung wurde Entwickler2 durch ein Gespräch außerhalb der Sitzung abgelenkt. Die dritte Pair-Developing-Sitzung zur Weiterentwicklung des Voice Recording-Feature dauerte 54 Minuten. Entwickler2 teilte anfangs kurz mit, welche Änderungen er zwischen den Sitzungen am Quellcode vorgenommen hatte. Danach fuhren sie mit der Implementierung fort und machten zwischendurch Ad-hoc-Tests auf dem Server. Am Ende fasste Entwickler1 die Sitzungen zusammen.
In der Sitzung zeigten sich neue technische Probleme. Beispielsweise wurde die Synchronisation von Saros automatisch abgebrochen. Des Weiteren hatte Entwickler1 Probleme mit seiner Netzwerkverbindung. Die Verbindung wurde mehrfach während der Sitzung unterbrochen.
Durch seine Angabe im Fragebogen nach der Sitzung machte mich Entwickler1 auf einen weiteren Konflikt bzgl. der Rollenverteilung aufmerksam: *"I think I spent too much time in driver role this session."*

Sitzung 9:
Entwickler2 verspätete sich 5 Minuten zur Reflexion vor der Sitzung. Dadurch und durch die Reflexion verzögerte sich der Sitzungsbeginn um 19 Minuten. Nach der Reflexion blieb Entwickler2 ungeplant und ohne Ankündigung als dritter Entwickler und stiller Beobachter in der Sitzung. Der Inhalt der Sitzung war ein weiteres Review von Entwickler3 für sein CDR-Feature. Entwickler3 hatte wieder Schwierigkeiten bei der Verwendung von Saros. Beispielsweise bemerkte er nicht, dass er einen anderen Benutzernamen als vorher verwendete und vergaß die kürzere Synchronisation mit einem existierenden Projekt zu wählen. Während der Synchronisation nannte Entwickler1 die Sitzungsziele und die Entwickler klärten erste Fragen. Nach einer verhältnismäßig kurzen Synchronisation von 3 Minuten stellte sich heraus, dass Entwickler3 eine falsche Version seines Quellcodes in die Versionsverwaltung eingecheckt hatte. Da es sich nur um wenige Abweichungen handelte, kopierte Entwickler1 diese in den bereits synchronisierten Quellcode. Während der Sitzung zeigte sich, dass die Entwickler sich auch dieses Mal nicht über das Vorgehen abgestimmt hatten. Entwickler3 fragte bspw., wie Entwickler1 gerne weiter verfahren würde oder welchen Teil des Quellcodes er als nächstes gerne sehen würde. Am Ende fasste Entwickler1 die Sitzung kurz zusammen.
Wieder traten technische Probleme mit der Netzwerkverbindung von Entwickler1 auf. Außerdem hatte Entwickler3 Probleme mit Saros, weil sein Cursor bei manchen Aktionen automatisch in das Fenster Share Project Session wechselte.

Sitzung 10:
Die mittlerweile vierte Sitzung vom Typ Pair-Developing wurde die erste offizielle Sitzung mit drei Entwicklern. Weder die Reflexion davor noch ein weiteres Saros-Update störten den Sitzungsfluss nennenswert. Entwickler3 war bisher in die Pair-Developing-Sitzungen nicht involviert gewesen. Während der Sitzung brachte er sich nicht ein, und am Ende war er ein weiterer stiller Beobachter gewesen. Als Entwickler1 in der Mitte der Sitzung ankündigte, dass er die Dri-

ver-Rolle abgeben wollte, reagierten die Entwickler aus Bangalore nur mit verlegenem Lachen, sodass kein Rollenwechsel stattfand. Zur Konfliktlösung bot ich an, dass sie den Multi-Driver-Mode verwenden könnten. Sie verwendeten ihn aber nicht, wie erwartet, zum parallelen Editieren. Entwickler2 nutzte die Gelegenheit, um sich im Quellcode unabhängig von Entwickler1 umzusehen. Nach wenigen Minuten signalisierte er jedoch, dass er nicht wusste, an welcher Stelle im Workspace Entwickler1 sich befinde, und folgte ihm danach wieder. Es zeigten sich neue technische Probleme wie z.B. fehlende Markierungen, die durch das neue Release verursacht wurden.

Sitzung 11:
Die 11. Sitzung war die erste Sitzung ohne technische Probleme. In der fünften Pair-Developing-Sitzung wurde wieder am Voice Recording-Feature gearbeitet. Dieses Mal war als Aufgabe geplant, die Stellen im Quellcode zu bearbeiten, die durch Fehlermeldungen abgefangen werden sollten. Entwickler1 erschien versehentlich 15 Minuten zu früh und Entwickler2 verspätete sich um 15 Minuten. Die Zeit dazwischen nutzte ich, um mit Entwickler1 ein Einzelgespräch über Maßnahmen und Erkenntnisse des Etablierungsprozesses zu führen. Entwickler3 war wieder eingeplant gewesen, konnte aber wegen Krankheit nicht erscheinen.
Entwickler1 lud wie immer in den Sitzungen als Host zur Synchronisation ein. Da die Synchronisation nur eine Minute dauerte, hatten die Entwickler am Ende noch Zeit, die Fragebogen (MRS) in der Sitzung auszufüllen. Entwickler2 nutzte die Gelegenheit, um weitere inhaltliche Fragen zu stellen, die ihm im Nachhinein noch einfielen. Wieder war Entwickler2 in der Sitzung von außen abgelenkt gewesen. Er bekam etwa nach der Hälfte der Zeit einen Anruf und telefonierte kurz. Trotzdem fiel im Gegensatz zu vorherigen Pair-Developing-Sitzungen auf, dass die Kommunikation zwischen den Entwicklern stabil war und ausreichend für Awareness gesorgt wurde. Entwickler1 kommentierte viele seiner Aktionen und Entwickler2 bestätigte durch „*mhm*“ oder stellte Rückfragen, wenn er etwas nicht sehen konnte.

4.7.3.3 Besondere Ereignisse und Probleme

In diesem Arbeitszyklus fanden zwei Sitzungen mit allen drei Entwicklern statt. Der dritte Entwickler beteiligte sich jedoch nicht an den jeweiligen Sitzungen, sondern verhielt sich wie ein stiller Beobachter.
Es war ebenfalls das erste Mal der Multi-Driver-Mode eingesetzt worden. Er wurde jedoch nicht, wie erwartet, zum parallelen Editieren eingesetzt, sondern bewirkte nur einen Awareness-Verlust des Observers.
Des Weiteren zeigten sich Konflikte über die Rollenverteilung in den Pair-Developing-Sitzungen. Entwickler1 erwartete, dass auch die anderen beiden Entwickler die Driver-Rolle teilweise übernehmen sollten, wozu sie aber nicht bereit waren.
In diesem Arbeitszyklus fand die erste Sitzung statt, deren Verlauf nicht durch technische Probleme unterbrochen wurde.

4.7.3.4 Ergebnisse des zweiten Arbeitszyklus

Auf die Informationen über den Standardsoftwareentwicklungsprozess des C5-Teams musste ich bis zum vierten Arbeitszyklus warten. Dafür gab es keinen speziellen Grund, außer vielleicht, dass diese Informationen weniger zeitkritisch waren als aktuelle Daten zu Sitzungen und ich daher mehr darum bemüht war, aktuelle Daten aus Fragebogen und Reflexionen zu erhalten.
Bisher fehlte dem Verlauf der Sitzungen aus meiner Sicht als Beobachterin immer noch die Struktur. Ich wies Entwickler1 in einem Einzelgespräch vor einer der Sitzungen darauf hin, dass die Sitzungen durch eine Einleitung während der Synchronisation und einer Zusammenfassung am Ende der Sitzung evtl. besser strukturiert werden könnten. Er kompensierte in den folgenden Sitzungen den Zeitaufwand während der Synchronisation, indem er über die Ziele und Aufgaben der folgenden Sitzung sprach.

Insgesamt war die Anzahl technischer Probleme innerhalb des Arbeitszyklus zurückgegangen. Die Vorbereitungsphase hatte sich in der 11.Sitzung auf eine Minute verkürzt. Auch die Audioverbindung hatte nach den Einstellungsänderungen vor der 8. Sitzung die Zusammenarbeit in den Sitzungen nicht weiter gestört. Dadurch blieb den Entwicklern mehr Zeit für die produktive Arbeit und die Erweiterung des Etablierungsprozesses wie z.B. Sitzungen mit drei Entwicklern oder der Einsatz einer weiteren Saros-Funktionalität (Multi-Driver-Mode).

Die Fragebogen halfen mir dabei, die Sitzungsinhalte und den Verlauf nachvollziehen zu können. Des Weiteren erhielt ich dadurch den ersten Hinweis auf mögliche Konflikte bzgl. der Rollenverteilung in den Sitzungen.
Die Entwickler hielten es weiterhin nicht für nötig, in den Reflexionen direkt miteinander zu kommunizieren. Es war den Entwicklern zwar unangenehm, aber sie beantworteten meine Fragen über Konflikte wie z.B. Ablenkungen von außen während der Sitzung oder die Rollenverteilung. Jedoch scheiterten Versuche, die Entwickler in der Reflexion dazu zu bringen, über die Konflikte selbstständig zu diskutieren.

4.7.4 Dritter Arbeitszyklus

4.7.4.1 Ziele und Maßnahmen für den dritten Arbeitszyklus

Ziel für den dritten Arbeitszyklus war unter anderem, die Kommunikation zwischen den Entwicklern in der Sitzung und der Reflexion zu erhöhen. Dies galt besonders im Bezug auf den Austausch von Erfahrungen und dem Umgang mit Problemen. Eine darüber verbesserte Arbeitsatmosphäre sollte offenes Feedback und somit schnelleren Umgang mit der Erkennung und Lösung von Konflikten ermöglichen.
Außerdem sollten den Entwicklern ihre Rollen, ihr Verhalten und ihre Ziele verdeutlicht werden, damit sie für prozessuale Aufgaben, z.B. Pünktlichkeit, klare Zielsetzung, aktive Teilnahme an den Sitzungen, mehr Verantwortung übernehmen würden.
Weiterer Bestandteil des dritten Arbeitszyklus war die Formulierung eines Textes über bisher gesammelte Daten und Erkenntnisse (auch in Form von Tabellen, Skizzen und Zeitverläufen) als Bestandteil des iterativen Prozesses der Forschungsmethode. Damit sollte für erste tiefere Analysen der Daten gesorgt werden. Darauf aufbauend sollten weitere Maßnahmen im Etablierungsprozess festgelegt werden.
Des Weiteren sollte eine Erhebung über den Zwischenstand des DPP-Projektes aus Sicht der Entwickler erstellt und ausgegeben werden.

1.	Direkte Kommunikation zwischen den Entwicklern erhöhen
2.	Bewusstsein der Entwickler für Rollen und Verantwortungen im DPP-Projekt erhöhen
3.	Erste Erkenntnisse über DPP durch Analyse des ersten Arbeitszyklus gewinnen
4.	Zwischenstanderhebung erstellen und durchführen

Tabelle 4.15 Maßnahmen im dritten Arbeitszyklus

4.7.4.2 Chronologische Beschreibung der Ereignisse in den Sitzungen

Sitzung 12:
In der mittlerweile sechsten Pair-Devoloping-Sitzung sollte eine weitere Funktionalität (Dialog Subscription) des Voice Recording-Features implementiert werden. Es sollte eine weitere Sitzung mit drei Entwicklern werden, doch Entwickler3 nahm nicht wie geplant teil. Zur Umsetzung wurde Quellcode eines anderen Wiener Teams, das für eine andere Anwendung eine ähnliche Funkti-

onalität geschrieben hatte, verwendet und angepasst. Da Entwickler1 im Fragebogen US angab: *„we both have to learn about dialog subscription“,* vermittelte er den Eindruck, dass der Wissensunterschied zwischen den Entwicklern im Gegensatz zu vorherigen Sitzungen nicht vorhanden war. Er hatte sich jedoch insoweit auf die Sitzungen vorbereitet, dass er schon Gespräche mit dem Autor geführt hatte und wusste, welche Stellen er überarbeiten wollte. Entwickler2 konnte sich durch den geringeren Wissensunterschied und seine Erfahrung als Entwickler besser in die Sitzungen einbringen. Dies drückte sich auch im positiven Feedback von Entwickler1 im Fragebogen MRS aus. Allgemein meinte er: *„found a lot of errors together“* und besonders für die Ausnahmebehandlung fand er DPP besonders hilfreich: *”found solutions really fast together“*.
Am Ende der Sitzung stellten die Entwickler fest, dass sie den Aufwand unterschätzt hatten und der vorhandene Code die Anforderungen bisher nur teilweise erfüllte. Da ungelöste Probleme vorhanden und Erweiterungen nötig waren, wurde die Beendigung der Aufgabe für die nächste Sitzung geplant.
Die Sitzung wurde durch Inkonsistenzen gestört, die wahrscheinlich durch Einstellungen bei der Synchronisation verursacht wurden. Entwickler1 hatte auf Anraten des Saros-Teams große Dateien, die die Entwickler lokal selbst aus dem Quellcode erzeugen konnten, bei der Synchronisation ausgeschlossen.

Sitzung 13:
In der siebten Pair-Developing-Sitzung sollten dann die Erweiterungen und die letzten Probleme aus der vorherigen Sitzung erledigt werden. Zwischen den Sitzungen hatte Entwickler1 sich mit dem ursprünglichen Autor des Codes beraten. Die Synchronisation war schnell erledigt, doch die Sitzung fing durch die Verspätung von Entwickler2 erst 17 Minuten später als geplant an. Es stellte sich im Verlauf der Sitzung heraus, dass Anpassungen in Verbindung mit einem Server (der in C++ geschrieben war) durchgeführt werden mussten, mit denen sich die Entwickler nicht gut genug auskannten. Da die Aufgabe nicht ohne Hilfe des ursprünglichen Autors gelöst werden konnte, wurde die Bearbeitung dieser Aufgabe von Entwickler1 abgebrochen. In der restlichen Sitzungszeit machte er spontan mit einer anderen Aufgabe weiter. Dafür suchte er sich eine Stelle, die mit einem „TODO“ markiert war, und fuhr dort mit der Implementierung fort.
Die Sitzung wurde schließlich durch eine Inkonsistenz beendet. Dieses Mal stellte sich heraus, dass Entwickler2 den Watchdog wahrscheinlich gedrückt hatte, bevor sich die Inkonsistenzen von selbst lösen konnten und dadurch ein Neustart der Saros-Sitzung nötig gewesen wäre.
Auch in dieser Sitzung drückte sich der geringere Wissensunterschied zwischen den Entwicklern, trotz der unerledigten Aufgabe, positiv in den Angaben von Entwickler1 im Fragebogen aus: *„we solved problems together. I think the interactivity between us is getting better.“*

4.7.4.3 Besondere Ereignisse und Probleme

In den Sitzungen dieses Arbeitszyklus arbeiteten die Entwickler mit fremdem Code. Dadurch wurde der Wissensunterschied zwischen ihnen kleiner. Jedoch konnten die Entwickler mit ihrem Wissen nicht alle Probleme innerhalb der Sitzung lösen und änderten daher spontan das Sitzungsziel.

4.7.4.4 Ergebnisse des dritten Arbeitszyklus

Die Kommunikation zwischen den Entwicklern wurde durch die Arbeit mit fremdem Code verbessert. Dadurch stellte sich eine neue Rollenverteilung ein, die eher *Average-Average* (siehe Abschnitt 2.3.1) entsprach und die es Entwickler2 ermöglichte, mehr von seinem Wissen in die Sitzungen einzubringen.
Des Weiteren konnte die Kommunikation durch zwei weitere Maßnahmen verbessert werden. Zum einen hielt ich mich aus der Eröffnung der Sitzungen heraus. Die Vorbereitungsphase hatte sich auf ein erträgliches Maß verkürzt und die Entwickler waren mittlerweile erfahrener im Um-

gang mit Saros, daher war an dieser Stelle keine Betreuung meinerseits mehr nötig. Zum anderen wurde den Entwicklern auch probeweise die Moderation der Reflexion übertragen. Meine Unterstützung beschränkte sich lediglich auf Themenvorschläge. Dies führte bspw. dazu, dass Entwickler1 Entwickler2 darum bat, nicht mehr zu spät zu Sitzungen zu kommen. Die Entwickler hielten die Reflexion jedoch weiterhin kurz. Es kam auch nicht zum erhofften Erfahrungsaustausch bzw. weitere meiner Themenvorschläge wurden einfach ignoriert.
Ein Einzelgespräch mit Entwickler1 gab mir weiteren Aufschluss über den Status von aktuellen Aufgaben und seine Erwartungen und Ziele an den weiteren Etablierungsprozess, z.B. auch welche Rolle Entwickler3 weiterhin haben würde, der zu dieser Zeit nicht an Sitzungen teilnahm.
Die Übertragung der Moderation in den Reflexionen und das Einzelgespräch mit Entwickler1 dienten unter anderem auch zur Erhöhung des Verantwortungsbewusstseins der Entwickler gegenüber dem Entwicklungs- und Etablierungsprozess.
Die ersten Versuche, die Erkenntnisse aus dem Etablierungsprozess schriftlich festzuhalten, offenbarten unter anderem, welche Informationen für die weitere Analyse fehlten und welche Ereignisse bisher besonders wichtig waren. Dadurch erhielt ich eine bessere Übersicht über den gesamten Etablierungsprozess und konnte gezielter auf Aspekte in Reflexionen eingehen.
Beim Entwurf der Zwischenerhebung stellte sich heraus, dass erst erörtert werden musste, mit welcher Befragungsform (Fragebogen, Einzelgespräch, Gruppengespräch, etc.) und unter Berücksichtigung welcher zusätzlichen Informationen (meine Beobachtungen, erste Analysen, etc.) die Erhebung am besten durchgeführt werden sollte (siehe Abschnitt 3.2 zur mündlichen Befragung in Einzelgesprächen). Daher wurde die Erhebung auf den vierten Arbeitszyklus verschoben.

4.7.5 Vierter Arbeitszyklus

4.7.5.1 Ziele und Maßnahmen für den vierten Arbeitszyklus

Ziel für den vierten Arbeitszyklus war unter anderem, die Kommunikation zwischen den Entwicklern in der Sitzung und in der Reflexion noch weiter zu verbessern.
Insgesamt sollten alle Maßnahmen, die mit dem Etablierungsprozess zu tun hatten (Fragebogen, Reflexion und Betreuung), angepasst werden, sodass die Entwickler mehr Verantwortung für die DPP-Sitzungen erhalten würden und gleichzeitig weniger Zeitaufwand für sie dadurch entstehen würde. Im Einzelnen war vorgesehen, (1) die Eröffnung der Sitzung und die Leitung der Reflexion ganz an die Entwickler abzugeben, (2) die Fragebogen zu kürzen und (3) die Betreuung in den Sitzungen weiter zu verringern.
Es sollte eine zweite Netviewer-Sitzung zu Entwickler2 erstellt werden, um seine Perspektive der Sitzungen besser nachvollziehen zu können und Verbesserungsvorschläge für eine aktivere Teilnahme seinerseits zu erforschen.
Entwickler2 sollte der Umgang mit dem Watchdog in Saros noch einmal verdeutlicht werden, um manche Unterbrechungen durch unnötige Neustarts von Saros in den Sitzungen zu vermeiden.
Auch in diesem Arbeitszyklus sollten im Rahmen des Forschungsprozesses weitere Analysen der gesammelten Daten durchgeführt werden. Damit sollten weitere Erkenntnisse über den bisherigen Verlauf gesammelt werden und in Gesprächen mit den Entwicklern thematisiert werden.
Des Weiteren sollte eine Zwischenerhebung in Form eines teilstrukturierten Interviews über die Bewertung der Entwickler bzgl. des DPP-Projektes durchgeführt werden.

1.	Direkte Kommunikation zwischen den Entwicklern in der Reflexion weiter verbessern
2.	Etablierungsprozess vereinfachen und Verantwortung an die Entwickler abgeben
3.	Betrachtung der DPP-Sitzungen aus Sicht von Entwickler2
4.	Weitere Erkenntnisse über DPP sammeln und Rückkopplung darüber in die Reflexionen
5.	Bewertung der Entwickler über das DPP-Projekt erheben

Tabelle 4.16 Maßnahmen im vierten Arbeitszyklus

4.7.5.2 Chronologische Beschreibung der Ereignisse in den Sitzungen

Sitzung 14:
Zwischen dieser und der letzten Sitzung war eine Pause von drei Wochen, weil Entwickler1 zuerst Urlaub hatte und danach sein Notebook defekt war. In der Zwischenzeit hatten sich neue Anforderungen für das Voice Recording-Feature ergeben, die Entwickler1 in die weitere Planung der Pair-Developing-Sitzung einbeziehen wollte. Erneut hatte einer der Entwickler (diesmal war es Entwicker2) das Saros-Update nicht rechtzeitig ausgeführt. Zu Sitzungsbeginn meinte Entwickler2, dass er die richtige Saros-Version installiert hatte. Nach der Synchronisation einige Minuten später stellte sich heraus, dass eine wichtige Saros-Funktionalität (der Verfolgermodus) nicht funktionierte, weil Entwickler2 doch nicht über die richtige Saros-Version verfügte. Technische Probleme (die Adresse des Updateservers war vom Büro in Bangalore nicht erreichbar) hielten Entwickler2 davon ab, Saros noch rechtzeitig für die Sitzung zu aktualisieren. Die weitere Sitzung bestand abwechselnd aus inhaltlichen Diskussionen zwischen den Entwicklern und verschiedenen Versuchen des Saros-Teams, eine Ad-hoc-Lösung für das Problem zu finden. Die nicht wie geplant vorhandene Desktop Sharing-Verbindung zu Entwickler2 hätte das Problem evtl. schneller beseitigt. Entwickler1 ließ sich durch die technischen Probleme nicht aus der Ruhe bringen und nutzte die Zeit zwischen der Problembehandlung immer wieder, um wenigstens die neuen bzw. geänderten Anforderungen mit Entwickler2 zu besprechen und sie in das weitere Vorgehen einzuplanen. Danach (die Sitzung dauerte bereits eine Stunde) brach Entwickler1 die Sitzung ab. Er gab im Fragebogen nach der Sitzung an, dass die erledigte Aufgabe, die wichtigere der beiden Sitzungsaufgaben gewesen war.

Sitzung 15:
Es waren vier Wochen vergangen, seit die Entwickler das letzte Mal gemeinsam in einer DPP-Sitzung gearbeitet hatten. Die achte Pair-Developing-Sitzung diente der Beendigung der Aufgabe aus Sitzung 13. Entwickler1 wollte keine weiteren Inkonsistenzen verursacht durch eine Teilsynchronisation riskieren. Daher ging er wieder dazu über, das gesamte Projekt zu synchronisieren. Dadurch verlängerte sich die Vorbereitungsphase wieder um einige Minuten. Dies hatte er jedoch eingeplant und eröffnete das Gespräch über die Entwicklung, während die Synchronisation lief. Zur technischen Betreuung der Sitzung war mittlerweile ein weiteres Mitglied des Saros-Teams als Beobachter an den Sitzungen beteiligt. Er nutzte die letzten Minuten der Synchronisation, um sich bei den Entwicklern vorzustellen.
Entwickler1 versuchte über die Position des Cursors und Nachfragen bei Entwickler2 zu erfahren, an welcher Stelle sich dieser im Workspace befand und ob er ihm folgen konnte. Während der Sitzung traten Inkonsistenzen auf, die sich aber von selbst auflösten. Entwickler2 war mehrmals von äußeren Einflüssen abgelenkt worden.

Sitzung 16:
Schon zu Sitzungsbeginn ließ mich Entwickler1 wissen, dass dies für die nächsten drei Wochen die letzte DPP-Sitzung sein würde, weil die Entwickler aus Bangalore für andere Aufgaben gebraucht wurden. Es handelte sich um eine weitere Pair-Developing-Sitzung, die wieder auf fremdem Quellcode basierte. Entwickler1 versuchte erneut, über die Position des Cursors von Entwickler2 und direkte Nachfrage zu ermitteln, worauf dieser gerade fokussiert war und ließ sich seine Vermutungen darüber bestätigen. Er verschob Anpassungen der Datenbank auf einen Zeitpunkt nach der Sitzung, weil Entwickler2 diese Aktivitäten außerhalb von Saros nicht verfolgen konnte.
Entwickler2 ließ sich wieder von äußeren Einflüssen ablenken. Er wurde von einer Kollegin angesprochen und klärte mit ihr in einer kurzen Unterhaltung eine von ihr gestellte Frage. Gegenüber Entwickler1 äußerte er sich dazu nicht. Entwickler1 reagierte souverän, indem er die Ablenkung beschrieb *„Somebody with questions next to you?“* und die Aufmerksamkeit sogleich durch Erklärungen über seine Aktionen wieder auf seine Aktivitäten lenkte.

4.7.5.3 Besondere Ereignisse und Probleme

In Sitzung 14 konnte Saros wegen technischer Probleme und zu kurzfristiger Vorbereitung von Entwickler2 nicht genutzt werden. Die Entwickler erledigten trotzdem eine der beiden Sitzungsaufgaben, die auch ohne Saros bearbeitet werden konnte.
Entwickler2 wirkte mehrfach in den Sitzungen durch äußere Einflüsse wie z.B. Telefonklingeln oder Fragen von Kollegen abgelenkt.

4.7.5.4 Ergebnisse des vierten Arbeitszyklus

In einem Einzelgespräch mit Entwickler1 bekam ich schließlich die Informationen über den Standardsoftwareentwicklungsprozess (siehe Abschnitt 4.4.4). Diese machten die inhaltliche Gestaltung der Sitzungen für mich leichter nachvollziehbar. Sie waren außerdem eine gute Grundlage für die weitere Analyse.
Wie geplant, wurden die Maßnahmen zur Erhöhung der direkten Kommunikation zwischen den Entwicklern und eine Reduktion des Aufwandes der Maßnahmen des Etablierungsprozesses durchgeführt (siehe Abschnitt 4.6). Im Zuge dessen wurde den Entwicklern endgültig die Reflexionsleitung (unterstützt durch Themenvorschläge meinerseits) überlassen.
Die Sitzungen wurden ab der 15. Sitzung durch einen weiteren Entwickler des Saros-Teams betreut, der an der Entwicklung von Saros beteiligt war und die technische Unterstützung der DPP-Sitzungen übernahm.
Die DPP-Sitzungen konnten nicht aus der Perspektive von Entwickler2 beobachtet werden, da die geplante Desktop Sharing-Sitzung bis zuletzt nicht mehr eingerichtet werden konnte.
Es traten im letzten Arbeitszyklus (außer in der 14. Sitzung) keine weiteren technischen Probleme mehr auf. Zum einen, weil Entwickler2 sich bei der Aktivierung des Watchdogs zurückhielt und zum anderen weil Entwickler1 alle Dateien in die Synchronisation einschloss und dadurch einige Inkonsistenzen nicht mehr auftraten.
Nachdem der Fokus nicht mehr nur auf technischen Problemen ruhte und Entwickler2 sich öfter von äußeren Einflüssen ablenken ließ, entwickelte sich besonders bei Entwickler1 ein stärkeres Bedürfnis danach zu wissen, dass Entwickler2 seine Aktivitäten verfolgte bzw. worauf dieser sich im Workspace fokussierte.
Aufgrund von Sitzungsausfällen wurde Sitzung 16 überraschend die letzte Sitzung, die in die Datenerhebung einfloss, und die geplante Zwischenerhebung wurde zu einer abschließenden Erhebung (siehe Abschnitt 5.1). Eine weitere Maßnahme, die Entwickler ohne Betreuung eine Sitzung durchführen zu lassen und im Anschluss ihre Erfahrungen daraus zu erheben, konnte daher auch nicht mehr umgesetzt werden.
Das DPP-Projekt war danach noch nicht beendet, doch die von mir betreuten Sitzungen endeten nach der 16. Sitzung, da die Entwickler aus Bangalore in den darauffolgenden Wochen keine Zeit mehr für DPP-Sitzungen hatten.

5 Analyse des Etablierungsprozesses und des Nutzens von DPP

Verschiedene Vorgehensweisen wurden zur Analyse der Daten und zur Beantwortung der Forschungsfragen verwendet.
Zuerst wurde eine abschließende Erhebung über den Etablierungsprozess aus Sicht der Entwickler durchgeführt, um zu erfahren, wie sie den Verlauf und die Ergebnisse bewerteten.
Zur Sammlung von genaueren Erkenntnissen über den Etablierungsprozess und DPP im Allgemeinen folgte danach eine tiefere Analyse des Etablierungsprozesses im Hinblick auf bestimmte Phänomene. In diesem Zusammenhang wurden sowohl Ursachen als auch Auswirkungen der Ereignisse ergründet.
Daraus resultierende Ergebnisse bildeten die Grundlage für die folgenden zwei Analysen. Zum einen wurden Erkenntnisse aus dem Etablierungsprozess zusammengefasst und daraus eine allgemeine Anleitung für Etablierungsprozesse abgeleitet. Zum anderen ließ sich daraus der Nutzen von DPP im industriellen Umfeld ableiten.

5.1 Abschließende Erhebung

Die abschließende Erhebung in Form eines teilstrukturierten Interviews sollte mir Aufschluss über die Bewertung der Entwickler bzgl. des DPP-Projektes geben. Da sie erste Erkenntnisse über den Etablierungsprozess aus Sicht der Entwickler präsentiert und nicht mehr als direkte Maßnahmen des Etablierungsprozesses diente, wurde sie nicht in die Beschreibung sondern in die Analyse übernommen.
Die Fragebogen nach den Sitzungen hatten bereits bestätigt, dass der Entwicklungsprozess, wie erwartet, verlaufen war und die Entwickler die geplanten Sitzungsziele bis auf wenige Ausnahmen erreichen konnten (siehe Abschnitt 5.3.2). Doch es fehlte noch der Gesamteindruck, den sie bezogen auf DPP und den Etablierungsprozess mittlerweile hatten.
Da Entwickler1 und Entwickler2 die meisten Sitzungen durchgeführt hatten, wurden sie in Einzelgesprächen dazu befragt (siehe Interviewleitfaden in Anhang C). Auf die Einstiegsfrage, die Erkenntnisse über Interessensschwerpunkte der Entwickler liefern sollte, gingen beide nicht weiter ein. Da die Entwickler, entsprechend den Erlebnissen in den Reflexionen, von sich aus keine Themen vorgaben, wählte ich also die Alternative mit ihnen direkt über ihre Ziele und Erfahrungen zu sprechen. Die Entwickler hatten Schwierigkeiten sich an konkrete Erfahrungen zu erinnern. Daher fragte ich sie zur Unterstützung nach konkreten Ereignissen aus den Sitzungen.

Ergebnisse der abschließenden Erhebung

Was den Entwicklern allgemein in Bezug auf den Etablierungsprozess aufgefallen war
Die Ziele der Entwickler wurden in Tabelle 4.6 bereits erläutert. Diese bestätigten sie auch in den Einzelgesprächen. Entwickler1 war positiv überrascht über seine Erfahrungen in Pair-Developing-Sitzungen. Es machte ihm mehr Spaß gemeinsam mit Entwickler2 Code zu produzieren und Lösungen zu finden. Im Gegensatz zu Reviews, bei denen er sich stark konzentrieren musste, weil er sehr schnell durch fremden Code durchgehen musste. Insgesamt bestätigte er noch einmal, dass das Projekt rechtzeitig zur Deadline fertig werden würde.
Entwickler2 äußerte sich vor allem positiv über Erfahrungen aus den letzten Sitzungen: "*So far so good. Especially last sessions… Right know it is going fine. I appreciate the sessions.*" Die Frage, ob es langweilig gewesen war ständig in der Observer-Rolle zu sein, beantwortete er, indem er die Situation aus seiner Sicht beschrieb: " *I feel sometimes like this, but the problem is that [Entwickler1] has developed the feature and he is in charge of the feature. What can I do? If I would have developed the feature I may have been in the driver role. I accepted that he has started and is the owner of the feature. I just observe and give my comments.*" Er akzeptiert die vorhandene Rollenverteilung, obwohl er sich vielleicht auch anderes gewünscht hätte. Auf die Frage, welche Erfahrungen er bzgl. der Sitzungen mit drei Entwicklern gemacht hatte, antwortete er (bezogen auf Sitzung 9): *„It was good. They were discussing, there was no role there, I was just observing"*

Insgesamt hielt er DPP-Sitzungen nicht für Zeitverschwendung, solange er keine Aufgaben zu erledigen hatte, die dringlicher waren.

Wie die Entwickler den Erfolg einer Sitzung beurteilten
Entwickler2 hatte eine eher produktorientierte Sicht. Er beurteilte eine erfolgreiche Sitzung danach, ob sie lauffähigen Code produziert hatte oder nicht. Wenn der Code nach der Sitzung noch nicht lauffähig war, versuchte er durch Testen und Korrekturen (bis maximal eine Stunde nach der Sitzung), das Sitzungsziel doch noch vollständig zu erreichen.
Für Entwickler1 war eine Sitzung erfolgreich, wenn darin Code erzeugt wurde oder Informationsaustausch, z.B. in Form von Diskussionen, die zu einem gemeinsamen Verständnis führten, zwischen den Entwicklern stattfand. Da er die Inhalte der Sitzung plante, konnte er gezielt auf gewünschte Ergebnisse hinarbeiten. Falls Sitzungsziele trotzdem nicht erreicht werden konnten, z.B. wegen technischer Probleme, versuchte er dies über weitere Gespräche zu klären.

Wie sich die Sicht der Entwickler auf DPP seit dem Beginn der DPP-Sitzungen (siehe Tabelle 4.3) verändert hatte
Entwickler1 wählte zuerst Reviews als Sitzungsaufgaben, weil er der Meinung war, dass sie sich für DPP-Sitzungen besser eignen würden. Als er später auch Pair-Developing-Sitzungen ausprobierte, war er positiv überrascht, dass auch dieser Sitzungstyp sich für DPP-Sitzungen eignet.
Entwickler2 hatte vor den Sitzungen DPP mit einer Definition beschrieben. Mittlerweile hat er eine genauere Vorstellung darüber. Er nannte zwei Aspekte, die für ihn neu waren. Zum einen konnte er sich nun eine bessere Vorstellung davon machen, was unter DPP zu verstehen ist. Zum anderen merkte er an, dass es für eine Zusammenarbeit wichtig ist, dass es sich um zwei gleichgesinnte Personen handelt, die gegenseitig ihre Gefühle verstehen können. Seine genauen Angaben waren: "*I think a lot of things coming into picture [...] It should be a session of two like-minded people. They understand each other's feelings well, then they can really work.*"

Übereinstimmungen und Widersprüche zwischen den Angaben der Entwickler
Die Entwickler gaben übereinstimmend an, dass sie sich nie über Projektziele, Sitzungsziele (ausgenommen konkrete Aufgaben) oder Angaben im Fragebogen ausgetauscht hatten.
Zur Begründung, warum die Entwickler sich nicht über ihre Ziele ausgetauscht hatten, antwortete Entwickler1: „*Weil Techniker nicht gerne über ihre Ziele reden oder planen wollen.*"
Außerhalb der Sitzungen kommunizierten sie meistens per Chat miteinander, z.B. um kurz vor der Sitzung die Aufgaben abzusprechen oder nach der Sitzung Probleme nach zu besprechen.
Im diesem Zusammenhang teilten beide übereinstimmend mit, dass sie keine expliziten Vorbereitungsmaßnahmen für die Sitzungen vereinbart hätten. Aus der Sicht von Entwickler1 wäre es besser gewesen, anhand von vollständigen Spezifikationsdokumenten das Projekt zu bearbeiten, da Sitzungsinhalte nicht so spontan gewesen wären und dadurch evtl. eine bessere Vorbereitung möglich gewesen wäre.
Widersprüchliche Angaben gab es in Bezug auf die Projektziele der Entwickler, wie schon in Abschnitt 4.4.2 erwähnt. Außerdem unterschieden sich ihre Meinungen über die Bevorzugung von Desktop Sharing im Gegensatz zu der vorhandenen technischen Infrastruktur mit Saros. Nach der Meinung von Entwickler2 waren Sitzungen, die keinen Code als Output lieferten, genauso gut per Desktop Sharing durchführbar. Entwickler1 hatte schlechte Erfahrungen mit der Verwendung von Desktop Sharing-Anwendungen gemacht, denn ungleiche Bildschirmauflösungen der Beteiligten hatten seine Erklärungsversuche in der Sitzung unverständlich gemacht. Seiner Meinung nach waren DPP-Sitzungen mit der vorhandenen technischen Infrastruktur (inkl. Saros) gegenüber Desktop Sharing-Anwendungen zu bevorzugen.

Zusammenfassung

Zusammenfassend kann man aus der abschließenden Erhebung folgern, dass die Entwickler die DPP-Sitzungen produktiv nutzen konnten und sich besonders in späteren Sitzungen bei der Anwendung von DPP wohlfühlten. Sie konnten geplante Aufgaben ausführen und meistens erreich-

ten sie ihre Ziele. Aus Sicht von Entwickler1 hätte der Etablierungsprozess davon profitieren können, wenn ihm zur Planung der Sitzungsinhalte vollständige Spezifikationsdokumente vorgelegen hätten.

5.2 Phänomene

Ein wichtiger Bestandteil der Auswertung der Daten aus dem Etablierungsprozesses war eine erste tiefere Analyse von konkreten Phänomenen. Die Vorgehensweise wurde bereits in Abschnitt 3.4 erläutert. Die Analyse diente als Grundlage für weitere Auswertungen bzgl. des Etablierungsprozesses und der Vor- und Nachteile von DPP im industriellen Umfeld. Weiterhin bietet sie einen genaueren Einblick und ein besseres Verständnis für Ereignisse und deren Zusammenhänge im Etablierungsprozess.
Die Erörterung der Phänomene wird durch eine Beschreibung der Problematik eingeleitet. Darauf folgt eine detaillierte Analyse über die Gründe, die Auswirkungen und andere situative Einflüsse (weiterhin als Kontext bezeichnet). Besonders wichtig ist die Tatsache, dass dabei immer versucht wurde, die Argumentation so gut wie möglich auf Daten zurückzuführen. Anschließend wird anhand einer Zusammenfassung der kausale Zusammenhang für Erkenntnisse über DPP in real existierenden Projekten hergestellt.
In den nächsten Abschnitten werden Phänomene zu fehlender Abstimmung von Zielen, Zeitverschwendung durch Ausfälle bzw. Verspätungen und Vorkommen und Fehlen von Awareness erörtert.

5.2.1 Fehlende Abstimmung von Zielen

Im Etablierungsprozess gab es drei verschiedene Ebenen von Zielen: Sitzungsziele, Projektziele und Prozessziele. Manche der Ziele konnten nicht oder nur teilweise erreicht werden, während sich andere unbewusst einstellten. Der folgende Abschnitt befasst sich mit der Erörterung der Gründe, Wechselwirkungen und Bedeutung fehlender Abstimmung über Ziele bei der Prozesseinführung von DPP.

Die verschiedenen Zielebenen lassen sich folgendermaßen beschreiben:

- **Projektziele** waren sitzungsübergreifend. Sie galten unabhängig von einzelnen Sitzungen. Eines der Projektziele war z.B. Funktionalitäten einer VoIP[13]-Software zu entwickeln. Wie Tabelle 4.6 schon zeigt, hatten die Entwickler auf Projektebene teilweise gleiche Ziele, teilweise aber auch nicht.
- **Prozessziele** galten sowohl für einzelne Sitzungen als auch für den gesamten Etablierungsprozess. Sie bezogen sich auf den Einsatz von DPP, bspw. eine geeignete Rollenverteilung im Projekt und den Sitzungen zu entwickeln.
- **Sitzungsziele** umfassten geplante Aufgaben für einzelne Sitzungen und konnten für jede Sitzung variieren. Ein Sitzungsziel war bspw.: *„Determine the cause for two exceptions. Fix the problems.“* Wenn sich die Sitzungsziele zwischen den Entwicklern widersprachen, konnte es vorkommen, dass dadurch der reibungslose Ablauf der Sitzung gestört wurde bzw. dass Sitzungsziele nur teilweise erreicht wurden.

Wie in Abschnitten 4.4.2 und 4.4.3 erläutert, hatte Entwickler1 als projekterfahrener Entwickler eine leitende Rolle und seine Projektziele waren sowohl der Wissenstransfer als auch Quellcode zu erzeugen (siehe Tabelle 4.6). Die neueingestellten Entwickler aus Bangalore wollten über das Erzeugen von Code ihr Domänenwissen erweitern. In den Sitzungen wurde an einem *Proof of Concept* bzw. einem Prototypen gearbeitet. Da für ein späteres Release vorgearbeitet wurde gab es nur eine vorläufige Spezifikation und etwas weniger Termindruck als üblich. Durch die DPP-

[13] VoIP (engl. Voice over Internet Protocoll) ist die Abkürzung für Telefonie über das Internet.

Sitzungen sollte erreicht werden, dass beide Abteilungen sowohl in Wien als auch in Bangalore die Entwicklung und die Wartung der bearbeiteten Features übernehmen können.

Für die Erörterung waren folgende Fragen relevant:

- Wessen Ziele werden betrachtet?
- Welche Zielebene ist betroffen?

Die Analyse wird zeigen, dass es fehlende Abstimmung von Zielen auf verschiedenen Zielebenen gab. Die Ziele wurden nach eigenen Angaben der Entwickler nie explizit zwischen ihnen abgesprochen (siehe Abschnitt 5.1). Lediglich konkrete Aufgaben wurden vor der Sitzung per Chat-Nachrichten mitgeteilt oder innerhalb von Sitzungen aufgeteilt. Trotzdem konnten die geplanten Aufgaben meistens umgesetzt werden und die gesetzten Ziele somit erreicht werden. Wie genau dies möglich war, wird anhand der folgenden Analyse aufgezeigt. Der Reihe nach werden nun Phänomene bzgl. Projekt-, Prozess- und Sitzungszielen untersucht. Wie sich bei der Erörterung der Phänomene in Abschnitt 5.2.2 herausstellte, wurden manche dieser Ziele durch Einwirkungen von außen, z.B. Entscheidungen eines Projektleiters oder Anforderungsänderungen, beeinflusst.

5.2.1.1 Nicht abgestimmte Projektziele

Nach den Angaben der beiden Entwickler in der abschließenden Erhebung bestand kein Zweifel mehr daran, dass die Entwickler von Anfang an keine deckungsgleichen Projektziele hatten. Entwickler1 gab an, dass sowohl die Erzeugung von Code, als auch das Vermitteln bzw. der Austausch von Wissen zu seinen Projektzielen gehörte. Beim Wissenstransfer ging es ihm primär um Erfahrungsaustausch und das Vermitteln von Wissen über die Domäne. Dies schloss auch Diskussionen über Entwurfsentscheidungen mit ein. Entwickler2 gab an, dass nur eine Sitzung, die lauffähigen Quellcode erzeugen würde, für ihn als erfolgreich angesehen werden würde. Wissenstransfer war keines seiner expliziten Ziele. Nur in 30 % der Sitzungen gab er in den Fragebogen Wissenstransfer als einen der wichtigsten Faktoren für den Erfolg der Sitzung an (siehe Frage 7 Fragebogen US-kurz in Anhang B).
Ein Grund für die Diskrepanz zwischen den Projektzielen war sicherlich, dass sie nicht explizit vereinbart wurden. Die Frage ist, wie sie trotzdem erreicht werden konnten und welche Auswirkungen die fehlende Abstimmung auf den Verlauf des Entwicklungs- und Etablierungsprozesses hatte.
Auch wenn Entwickler2 Wissenstransfer nicht explizit als Projektziel angab, zeigte sich, dass dieser sich wenn nötig von selbst einstellte (vgl. *Pair-Learning* Abschnitt 2.1). Beispielsweise gab er in der 5. Sitzung im Fragebogen vor der Sitzung als einen der Vorteile von DPP für die Sitzung an:*„Knowledge of the feature going to be developed“* (siehe Abschnitt 4.7.2). Daher hatte die fehlende Abstimmung der Ziele in diesem Fall keine weiteren Auswirkungen auf den Entwicklungs- bzw. Etablierungsprozess.
Es stellte sich demgegenüber als wesentlich problematischer heraus, dass auch prozessuale Ziele nicht untereinander abgestimmt wurden.

5.2.1.2 Nicht abgestimmte Prozessziele

Entwickler1 hatte sich für die Sitzungen kein explizites Lernkonzept überlegt, mit dessen Hilfe er die neuen Entwickler über die Domäne informieren wollte. Er ging davon aus, dass sich durch die Aufgabenverteilung und die geplanten Sitzungsaufgaben alle weiteren prozessualen Arrangements von selbst einstellen würden. Wie auch andere Untersuchungen bestätigen, ist es jedoch deutlich schwieriger DPP ganz ohne gemeinsame Regeln anzuwenden [CCV03, Yap05].
In der 5. Sitzung wurde bereits deutlich, dass die Aufgabenverteilung im Projekt die Diskrepanz zwischen den Vorstellungen der Entwickler über die prozessuale Vorgehensweise noch erhöhte. Dies lässt sich wie folgt erklären. Für die Zusammenarbeit zwischen Entwickler1 und Entwick-

ler2 waren zwei verschiedene Projektaufgaben vorgesehen (siehe Tabelle 4.7). Erstens sollte Entwickler2 ein eigenes Modul erzeugen und weiterentwickeln. Ein Review darüber war Inhalt der ersten produktiven Sitzung. Zweitens sollte ab der 5. Sitzung der Grundstein für die weitere Zusammenarbeit in Pair-Developing-Sitzungen gelegt werden (siehe Abschnitt 4.5.2).
Entwickler1 gab im Fragebogen als geplante Sitzungsaufgaben an: „*First we have a look on the existing code. Then we start to implement a voice recording solution*". Entwickler2 gab an: *"New task discussion, design discussion, may be code later ... Feature is prototyped by [Entwickler1] and going to be completed by me"*.
Die Beschreibungen der geplanten Aufgaben der beiden Entwickler deckten sich weitestgehend. Ein Detail sorgte jedoch später für Konflikte. Denn Entwickler2 ging von Anfang an von getrennten Verantwortlichkeiten aus. Seine Formulierung schloss eine gemeinsame Verantwortung aus. Er erwartete, dass die Verantwortung über die Weiterentwicklung von Entwickler1 mit dieser Sitzung auf ihn übergehen würde. Dahingegen wollte Entwickler1, wie bereits in Abschnitt 4.5.2 erörtert, eher eine weitere XP Praktik einführen, die sowohl ihn als auch Entwickler2 dazu befähigen würde den existierenden Quellcode zu verändern (*Collective Ownership* [Bec99]).

Die Bearbeitung des SNMP-Features in der ersten Review-Sitzung hatte die Entwickler noch nicht in die Verlegenheit gebracht über Rollen und Verantwortlichkeiten nachdenken zu müssen, denn die Verantwortlichkeiten waren klar getrennt. Entwickler2 war Autor und zuständig für die Entwicklung. Er führte als Driver durch den von ihm erzeugten Code, während sie gemeinsam offene Entwurfsfragen klärten. Erst die zweite Aufgabe (Weiterentwicklung des Voice Recording-Features) offenbarte die Notwendigkeit für ein gemeinsames Verständnis der prozessualen Vorgehensweise.
Entwickler1 präsentierte Entwickler2 den vorhandenen Code. Er hatte jedoch nicht explizit geäußert, wie er sich die Rolle von Entwickler2 beim Pair-Developing vorstellte oder welche Erwartungen er an diesen hatte. Weder hatten sie geklärt, welche Aufgaben Entwickler2 übernehmen sollte, noch welche Verantwortungen er in den Sitzungen hatte. Bis zum Projektende behielt Entwickler2 daher seine Vorstellung von getrennten Verantwortlichkeiten und fühlte sich in den Sitzungen nicht verantwortlich für die Entwicklung des Codes. Er blieb beim Pair-Developing meistens Observer, gab wenige Kommentare oder stellte Verständnisfragen und ließ Entwickler1 als Driver die Implementierung durchführen (siehe Abschnitt 4.5.2).
Erst nach der 16. Sitzung stellte sich in einem Einzelgespräch mit Entwickler2 heraus, dass er entgegen der Idee von *Collective Ownership* davon ausging, dass nur der ursprüngliche Autor eines Dokumentes Driver sein sollte und mit der Driver-Rolle auch die Verantwortlichkeit für den Code einherging. Offensichtlich hatte er sich vorher nicht darüber geäußert, weil er es als eine Tatsache akzeptiert hatte (siehe Abschnitt 5.1).

Wie bereits im Abschnitt 2.3.1 über Pair-Developing erläutert, entsprach dies einer typischen Rollenverteilung von *Expert-Novice*. Auch in anderen Untersuchungen wurde festgestellt, dass es bei dieser Kombination von Erfahrungsstufen selten Rollenwechsel stattfanden, der Experte meistens die Driver-Rolle einnahm und 85 % der Zeit redete [SL08]. Dies wurde jedoch nicht als negatives Ergebnis eingestuft, da sowohl der Novize als auch der Experte davon profitierten.
Auf die Fertigstellung der Sitzungsaufgaben in Pair-Developing-Sitzungen hatte die fehlende Abstimmung der Prozessziele keine Auswirkungen. Denn aufgrund der Projekterfahrung von Entwickler1 fiel die zumeist passive Rolle von Entwickler2, gemessen am Code-Output am Ende der Sitzungen, nicht auf. Falls es doch noch unerledigte Aufgaben gab, wurden diese zwischen den Entwicklern in getrennter Verantwortlichkeit außerhalb der Sitzungen zu Ende gebracht. Hierbei übernahm auch Entwickler2 eine aktive Rolle, da es klar definierte, getrennte Verantwortlichkeiten gab.
Doch zeigte sich während des Etablierungsprozesses zunehmend, dass sowohl Entwickler1 als auch Entwickler2 mit den fehlenden prozessualen Arrangements ihre Schwierigkeiten hatten und sich dadurch Unzufriedenheit einstellte. Die fehlende Abstimmung der prozessualen Ziele und das unterschiedliche Rollenverständnis werden auch bei der Analyse zu Verspätungen und Awareness in den Abschnitten 5.2.3 und 5.2.4 nochmals aufgegriffen.

5.2.1.3 Phänomene zu Sitzungszielen

Wandlungen und Widersprüche der Sitzungsziele von Entwickler2

Vor jeder Sitzung gaben die Entwickler im Fragebogen US an, was ihnen besonders wichtig für die Erreichung der Ziele erschien (siehe Frage 7 US in Anhang B). Im Verlauf des Projektes kehrten sich manche Prioritäten in ihr Gegenteil um. Sowohl bei Entwickler1 als auch bei Entwickler2 gab es hinsichtlich der Sitzungsziele Veränderungen. So war für beide Entwickler besonders am Anfang wichtig, dass keine technischen Probleme die Erreichung der Ziele verhindern.
Die Sitzungsziele von Entwickler2 veränderten sich während des Etablierungsprozesses auch bezogen auf andere Prioritäten signifikant. Abbildung 5.1 zeigt, welche Relevanz die pünktliche Beendigung der Sitzungsaufgabe im Vergleich zur Relevanz für den Erhalt aller wichtigen Informationen für Entwickler2 jeweils hatte.

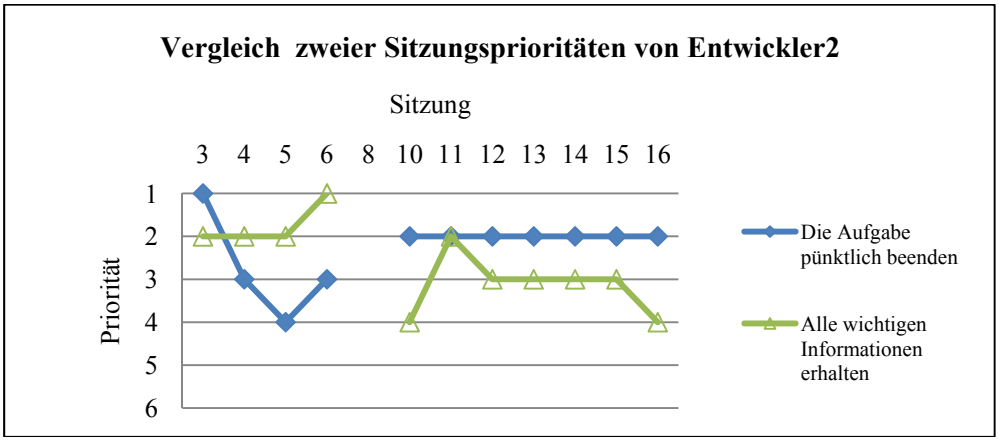


Abbildung 5.1 Angaben von Entwickler2 über die Prioritäten die Sitzungsaufgaben pünktlich zu erledigen und alle wichtigen Informationen zu erhalten (1 = sehr wichtig bis 6 = am wenigsten wichtig)

Wie Abbildung 5.1 zeigt, gab Entwickler2 dem Erhalt von Informationen besonders in der ersten Hälfte des Projektes eine hohe Priorität, während in der zweiten Hälfte vor allem die pünktliche Beendigung der Sitzungsaufgabe für ihn einen wichtigen Faktor für den Sitzungserfolg darstellte.

Ein Grund für die Wandlung wurde von Entwickler2 nicht genannt. Es lässt sich jedoch aus dem Kontext des Etablierungsprozesses vermuten, dass auch hier die Inhalte bzw. die Verantwortlichkeiten in Sitzungen eine Rolle spielten. In der 3. Sitzung hatte Entwickler2 selbst den Code erstellt, er hatte als Driver und Autor relativ viel Verantwortung für den Sitzungsablauf. Das Bedürfnis Feedback von Entwickler1 zu bekommen war hoch. Die Aufgabe (seinen Code zu präsentieren) pünktlich zu beenden ebenfalls.
In der 5. und 6. Sitzung wurde mit dem Pair-Developing ein neuer Sitzungstyp eingeführt. Entwickler2 wusste noch nicht, was auf ihn zukommen würde und daher lag seine Priorität eher darauf weitere Informationen zu erhalten, als pünktlich fertig zu werden. Als sich später herausstellte, dass die Pair-Developing-Sitzungen weitestgehend in der Verantwortung von Entwickler1 durchgeführt wurden, bekam die Priorität pünktlich fertig zu werden wieder einen höheren Stellenwert.
Diese Entwicklung ist nicht ungewöhnlich. Verschiedene Studien über DPP führten an, dass für die Entwickler vor allem die Entwurfsentscheidungen am Anfang eines Projektes relevant sind und sie in späteren Implementierungsphasen auch gerne getrennt weiterarbeiten [CCV03, WK00].

Entsprechend der Angaben im Fragebogen hätte es Entwickler2 besonders in der zweiten Projekthälfte wichtig sein müssen, die gesamte Sitzungszeit für die Erledigung der Aufgabe zur Verfügung zu haben. Wie sich bei der Erörterung zu Verspätungen in Abschnitt 5.2.3 zeigte, hielt ihn dies jedoch nicht davon ab regelmäßig zu spät zu Sitzungen zu erscheinen.

Widersprüchliche Sitzungsziele zwischen den Entwicklern

Die Entwickler hatten mehrfach in Sitzungen unterschiedliche Sitzungsziele angegeben. Besonders anschaulich war es jedoch in der 7. Sitzung (siehe Abschnitt 4.7.2).
Der Typ der Aufgabe in der Sitzung war von beiden Entwicklern im Fragebogen als Review deklariert worden. Entwickler1 schrieb als Sitzungsziel: *„Get a good knowledge of the […] code“*. Während Entwickler3 angab: *„Mostly knowledge and experience sharing, clarifying the approach before it's too late.“* Entwickler1 hatte ein Review geplant, um den erzeugten Code besser kennenzulernen. Entwickler3 hatte die Intention von Entwickler1 überprüfen zu lassen, ob er die vorläufige Spezifikation richtig umgesetzt hatte. Soweit konnte man noch keine großen Differenzen feststellen. Doch weitere Angaben im Fragebogen zeigten, dass ihre Vorstellung der Umsetzung sehr verschieden war. Zur Vorgehensweise schrieb Entwickler1: *„Synchronizing the Projects. Code Review presented by [Entwickel3] and [Entwickler2] followed by a discussion.*“ Entwickler3 schrieb hingegen: *„Get the code reviewed by the reviewer and discuss the open issues if any.“*
Die Widersprüche in den Sitzungszielen wirkten sich sowohl auf den Ablauf der Sitzung als auch auf die Erreichung der Ziele aus. Wie es dazu kam, wird nun im Detail erläutert. Der Grund für die uneinheitliche Vorstellung über die Umsetzung eines Reviews war, dass diese nicht zur üblichen Praxis der Entwickler in ihren Projekten gehörte. Normalerweise wurde der erzeugte Code erst nach Fertigstellung in der Testphase vom Testteam überprüft.
Durch die Angaben der Entwickler wird bereits deutlich, dass Entwickler1 erwartete, dass ihm der Code von den anderen Entwicklern präsentiert werden würde. Im Gegensatz dazu ging Entwickler3 davon aus, dass nicht er als Autor des Codes, sondern Entwickler1 die aktive Rolle übernehmen würde, wie es in Reviews üblich ist.[14] Da die Entwickler sich nie über Sitzungsziele austauschten, erfuhr Entwickler3 erst zu Sitzungsbeginn, welche Rolle er in der Sitzung hatte. Entwickler1 formulierte es folgendermaßen: *„The goal is to get a short intro into the code. You can open every class you want and you can highlight.“* Darauf reagierte Entwickler3 wie folgt: "*I will give you the status of how far we are and where we are stuck."*
Entwickler3 ließ sich nicht anmerken, dass er eine andere Vorstellung über den Ablauf hatte. Er ging spontan auf die Erwartungen und die Aufforderung von Entwickler1 ein. Ohne die Angaben im Fragebogen wären die Widersprüche zunächst nicht aufgefallen.
Doch spätestens bei der Präsentation zeigte sich, dass Entwickler3 nicht davon ausgegangen war die aktive Rolle im Review zu haben. Beispielsweise stockte er mehrfach bei seinen Erklärungen (siehe Tabelle 5.1b) und c)) oder ihm fiel mitten in einer Erklärung auf, dass er wichtige Punkte ausgelassen hatte (siehe Tabelle 5.1a), b) und c)), sodass er seine Erklärung unterbrach, um an einem anderen Punkt fortzufahren.

a)	*"Maybe it is better to start with [Dateiname]"*
b)	*„Here you get all the parameters … oh sorry, before that I need to explain how it gets the parameters.“*
c)	*"It is slightly difficult to find a starting point …ok…here you can see…ok…I also forgot to mention that […]"*

Tabelle 5.1 Erklärungsversuche von Entwickler3 bei einer Codepräsentation in der 7. Sitzung

Die Tatsache, dass Entwickler3 entgegen den Empfehlungen für Reviews als Autor selbst durch den Code führte und die unstrukturierte Vorgehensweise, können evtl. für zwei weitere Auswirkungen verantwortlich sein. Erstens ist es möglich, dass Fehlinterpretationen der vorläufigen Spezifikation von Entwickler1 unentdeckt blieben, weil Entwickler3 nur die Punkte ansprach, die er für kritisch hielt. Zweitens schien es aus demselben Grund und der fehlenden Vorbereitung von Entwickler1 völlig dem Zufall überlassen Defekte aufzuspüren.

[14] Vgl. Abschnitt 2.3.2

Sitzungszieländerung

Ungewöhnlich für den bisherigen Etablierungsprozess war die Zieländerung, die in der 13. Sitzung auftrat, nachdem die Entwickler feststellten, dass sie die geplanten Aufgaben nicht ohne fremde Hilfe erledigen konnten. Wie bereits in Abschnitt 4.7.4 ausführlich beschrieben, war es nach der 12. Sitzung bereits die zweite Sitzung, in der diese Aufgabe bearbeitet wurde und als die Entwickler nicht mehr weiter wussten, suchte Entwickler1 sich eine andere TODO-Markierung im Code und fuhr damit fort.
Der Wissensunterschied zwischen den Entwicklern war in diesen Sitzungen kleiner, weil sie ausnahmsweise mit Code arbeiteten, der auch für Entwickler1 neu war. Doch vielleicht lag gerade an dieser Stelle das Problem. Das Wissen der Entwickler reichte nicht aus, um das Problem ohne Hilfe von außen zu lösen. Obwohl Entwickler1 vor der Sitzung mit dem Autor des Codes über Lösungsansätze gesprochen hatte, konnten die Entwickler die geplante Aufgabe nicht beenden. Dies merkte er auch im Fragebogen zur Sitzung an: *"changed target of session during the session since it is easier to ask developers of 3rd party software why they send an error"*.
Die Frage ist, warum Entwickler1 die Sitzung nicht verlassen hatte, um das Problem sofort zu klären. Lag es vielleicht an der Sitzungssituation? Hatte ihn vielleicht die technische Infrastruktur oder die kurze gemeinsame Sitzungszeit davon abgehalten? Oder war die Aufgabe so komplex, dass Entwickler1 sich nicht vorstellen konnte, dass das Problem durch eine kurze Rückfrage gelöst werden könnte? Was hatte ihn also dazu bewogen mit einer anderen Aufgabe fortzufahren, statt sich Hilfe bei seinem Kollegen zu suchen? Im Fragebogen gab er folgenden Grund an: *" I can walk over to the next room to ask the developer. I think it wouldn't be a good idea to leave the session to do this"*
Diese Aussage macht den Eindruck, dass Entwickler1 die Zieländerung durchführte, weil er die Sitzung nicht verlassen wollte. Doch auf Nachfrage, gab er in einer Email folgende Begründung an: *„Das macht jeder wie er will. Mich stört es immer, wenn ich permanent unterbrochen werde, deshalb versuche ich die Kollegen zu erwischen, wenn sie nicht vor dem Computer sitzen (z.B. beim Essen, Rauchen, Kaffee trinken)."*
Er wechselte also die Aufgabe in der Sitzung, weil er seinen Kollegen nicht bei dessen Tätigkeiten stören wollte und bestätigte damit, dass die Sitzungssituation nicht der Grund dafür war.
Beim Mittagessen konnte er, wie erwartet, das Problem mit Hilfe seines Kollegen lösen und so konnten die letzten Probleme durch eine kleine Anpassung am Code gelöst werden.
Die Zieländerung hatte also weitere negative Auswirkungen auf die Sitzung verhindert. Entwickler1 konnte durch sein flexibles Handeln trotz fehlender Informationen durch die Zieländerung die restliche Sitzungszeit sinnvoll nutzen. Dies hatten ihm auch seine vorher gesetzten TODO-Markierungen ermöglicht.
Diese Situation zeigte zwei Aspekte von DPP. Zum einen wurde deutlich, dass fehlende Informationen den Sitzungsverlauf verändern können und flexible Handlungsweisen erforderlich sind, um negative Auswirkungen zu verhindern. Zum anderen bestätigte es den in Abschnitt 2.2 erwähnten Nachteil von DPP, dass gemeinsame Pausen für Entwickler1 einen wichtigen Bestandteil zum Informationsaustausch darstellten, der im virtuellen Team nicht möglich war.

5.2.1.4 Zusammenfassung

Aus der Analyse wird deutlich, dass eine fehlende Abstimmung der Ziele in vielerlei Hinsicht Einfluss auf den Nutzen von DPP haben kann.
Wenn die Entwickler nicht über ihre Prozessziele sprechen, kann es passieren, dass bis zuletzt die Vorstellungen über die Rollenverteilung und damit zusammenhängende Verantwortungen und Erwartungen voneinander abweichen. Dies kann sich auch auf die Prioritäten von Sitzungszielen und die aktive Unterstützung des zweiten Entwicklers in den Sitzungen negativ auswirken.

Fehlende Kommunikation kann bei DPP dadurch entstehen, dass Entwickler, die daran gewöhnt sind den Informationsaustausch auf ihre gemeinsamen Pausen zu beschränken, sich noch an die verteilte Situation anpassen müssen.

Verschiedene Projektziele und sich wandelnde Sitzungsziele müssen keine negativen Auswirkungen auf die Erledigung der Aufgaben haben. Solange die Vorstellungen über das gemeinsame Ziel, z.B. Code-Output erzeugen, nicht zu weit voneinander abweichen und die abweichenden Ziele, z.B. Wissenstransfer, implizit im Prozess enthalten sind.
Eine fehlende Abstimmung über die Sitzungsziele und deren Umsetzung kann zu Missverständnissen über Erwartungen und Verantwortlichkeiten führen. Negative Auswirkungen auf den Sitzungsablauf können teilweise durch flexibles Handeln verhindert werden. Allerdings resultiert dies dennoch in einer unstrukturierten Vorgehensweise.

5.2.2 Ausgefallen

Die Analyse der folgenden zwei Abschnitte (5.2.2 und 5.2.3) befasst sich mit einigen Phänomenen, die dazu führten, dass im DPP-Projekt Zeit verschwendet wurde. Manche der genannten Probleme, z.B. fehlende Vorbereitung, konnten darin nur am Rand erörtert werden und bleiben zukünftigen Untersuchungen überlassen.
Im Verlauf des DPP-Projektes kam es zu mehreren Sitzungsausfällen, weshalb von den ursprünglich angedachten 22 Sitzungen[15] insgesamt nur 16 Sitzungen stattfanden. Im folgenden Abschnitt sollen die Gründe, Wechselwirkungen und Bedeutungen von Sitzungsausfällen für eine Prozesseinführung von DPP erläutert werden.

Es gab verschiedene Gründe, die dazu führten, dass Entwickler der Sitzung fern blieben oder Sitzungen verschoben werden mussten. Sie werden folgendermaßen kategorisiert:

- Busy: Sitzungen wurden von den Entwicklern verschoben, weil dringendere Aufgaben zu erledigen waren.
- No-Show: Es kam mehrmals vor, dass Entwickler nicht wie geplant zu Sitzungen erschienen.
- Out-of-Office: Verschiedene Beteiligte konnten wegen spontanem Urlaub oder Krankheit nicht an Sitzungen teilnehmen.
- Fehlende Voraussetzungen/technische Probleme: Sowohl technische Probleme als auch fehlende Vorbereitung der Entwickler führten zu Sitzungsausfällen.

Bei der Analyse zeigte sich, dass folgende Fragen für die Erörterung eine Rolle spielten:

- Wer hat abgesagt?
- Wer hat entschieden, dass die Sitzung ausfällt?
- Wie und wann wurde es kommuniziert?

Im restlichen Teil dieses Abschnitts wird näher erörtert unter welchen Bedingungen Ausfälle zustande kamen und welche Auswirkungen dies auf das DPP-Projekt hatte. Die Erörterung geht dabei nach dem Verursacher der Ausfälle vor: (1) Entwickler1, (2) Entwickler2 oder Entwickler3, (3) fehlende Betreuung und (4) technische Probleme. Es ist leicht bei den vielen Vorfällen den Überblick zu verlieren, daher befindet sich in Tabelle 5.14 eine Übersicht über alle Ausfälle. Diese Übersicht wird während der Erörterung schrittweise aufgebaut.

[15] Dies entspricht einer Sitzung pro Woche.

5.2.2.1 Sitzungsausfall durch Entwickler1

Busy I

An dem DPP-Projekt waren drei Entwickler beteiligt. Die Planung und die Zielsetzung erforderten jedoch, dass Entwickler1 in jeder Sitzung anwesend war. Wenn Entwickler1 Sitzungen absagte, gab es keine alternative Besetzung, daher bedeutete es in jedem Fall, dass die Sitzung ausfallen würde. Drei Sitzungen mussten deswegen verschoben werden (siehe Tabelle 5.2).

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
4	1 Tag vorher per Email	*"Since I am very busy with urgent tasks I have to cancel the session this week. We will continue next week."*
9	1 Tag vorher in der Reflexion	*„According the session tomorrow, I have to cancel, because I have a Meeting at 10:30."*
13	1 h vorher per Email	*"Due to a needed emergency Patch for a customer I have to cancel the session today."*

Tabelle 5.2 Informationen zu Sitzungsausfällen, die durch Entwickler1 verursacht wurden

Während des gesamten Projektes war das Zustandekommen von geplanten DPP-Sitzungen unsicher, weil den Entwicklern jederzeit eine dringendere Aufgabe dazwischen kommen konnte. Entwickler1 ließ drei Sitzungen ausfallen, weil andere Aufgaben mit einer höheren Priorität wie z.B. ein Meeting oder dringende Wartungen für Kunden zu erledigen waren.
Nach Angaben der Entwickler wurde außerhalb der DPP-Sitzungen nicht an den Aufgaben des DPP-Projektes weitergearbeitet. Die Beendigung der Aufgaben wurde also durch die Ausfälle der Sitzungen um mehrere Wochen verzögert. Mehrmals versuchte Entwickler1 diese Ausfälle durch Erhöhung der Anzahl der Sitzungen pro Woche auszugleichen. Sitzung 8/9, 9/10, 12/13 und 15/16 waren in der gleichen Woche geplant. Für zwei der vier Wochen funktionierte es. Sitzung 9 und 13 wurden, wie vorher erwähnt, von Entwickler1 selbst kurzfristig abgesagt. Es stellte sich heraus, dass es für die Entwickler schwierig war spontan zusätzliche Sitzungen an nicht Sitzungstagen durchzuführen.
Durch die kurzfristigen Absagen wurden auch die Zeitpläne aller anderen Beteiligten gestört. Eine Absage wie in der 13. Sitzung, die erst eine Stunde vorher mitgeteilt wurde, blockierte den Zeitraum gegenüber anderen Terminen, die zu dieser Zeit hätten stattfinden können. Weiterhin hatten die Beteiligten ihre Tagesplanung darauf ausgerichtet und evtl. bereits unnötig Zeit für die Vorbereitung auf die Sitzung aufgewendet.

Out-of-Office

Wie in jedem Arbeitsalltag, gab es auch in den Sitzungen Ausfälle wegen Krankheit und Urlaubszeiten. Die 14. Sitzung fiel aus, weil Entwickler1 Urlaub hatte.
Natürlich könnte man sich jetzt fragen, warum die Sitzung ausfiel und nicht früher eine Alternative geplant worden war. Dies lag an der Organisation der Sitzungsplanung. Sitzungen wurden ausschließlich von Entwickler1 geplant. Er beurteilte, wann es nötig war Sitzungen durchzuführen. Von Anfang an hatte er Sitzungen nur nach Bedarf von Woche zu Woche geplant. Da das Projekt bis dahin gut in der Zeit lag, war wohl seiner Meinung nach keine Alternative nötig.
Was er zu diesem Zeitpunkt noch nicht wissen konnte war, dass die 14. Sitzung danach noch zwei weitere Male ausfallen würde.

Sitzungs-nummer	Abgesagt durch	Grund	Zeitpunkt und Art der Absage	Verschoben um	Probleme / Auswirkungen
4	Entwickler1	dringendere Aufgabe	1 Tag vorher per Email	1 Woche	unbeständige Planung,
9	Entwickler1	Anderes Meeting	1 Tag vorher in der Reflexion	1 Tag	Projektverzögerung
13	Entwickler1	dringendere Aufgaben	1 h vorher per Email	1 Woche	unbeständige Planung,
14	Entwickler 1	Urlaub	1 Woche vorher in der Sitzung	1 Woche	kurzfristige Planung, Projekt-verzögerung

Tabelle 5.3 Zusammenfassung der Sitzungsausfälle verursacht durch Entwickler1 wegen dringenderer Aufgaben

5.2.2.2 Ausfall von Entwickler2 bzw. Entwickler3

Neben Entwickler1 musste mindestens einer der beiden Entwickler aus Bangalore (Entwickler2 oder Entwickler3) anwesend sein, damit eine DPP-Sitzung stattfinden konnte. In diesem Abschnitt wird erörtert, warum das Fernbleiben der Entwickler aus Bangalore zwar nicht immer zu Sitzungsausfällen führte, sich aber trotzdem negativ auf die Sitzungen auswirkte.

No-Show I

Entwickler2 und Entwickler3 waren in der 2. Sitzung für den Einführungsprozess eingeplant gewesen. Doch nur Entwickler2 erschien zu dem verabredeten Termin und die Sitzung fand ohne Entwickler3 statt.
Der Grund des Nichterscheinens blieb ungeklärt. Es wurde weder von Entwickler3 noch von den anderen Beteiligten zur Sprache gebracht, warum er der Sitzung fernblieb.
Dadurch verpasste Entwickler3 sowohl erste Erfahrungen mit DPP, als auch die Möglichkeit die technische Infrastruktur (inkl. Saros) einmal auszuprobieren.
In Sitzungen mit seiner Beteiligung zeigte sich später, dass ihm essentielles Wissen über Saros fehlte. Er wählte bspw. in einer dieser Sitzungen die Option das gesamte Projekt zu synchronisieren, statt eine Teilsynchronisation durchzuführen, was zu einer Verzögerung von bis zu 30 Minuten (also ein Drittel der gesamten Sitzungszeit) führte. Da ein Zeitpuffer für die Vorbereitung eingeplant war, hatte es jedoch keine Auswirkungen auf die Erreichung der Sitzungsziele.

No-Show II

Auch Entwickler2 erschien nicht zu einer Sitzung, obwohl er dafür eingeplant war (siehe Tabelle 5.4).

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
7	In der Sitzung	*„I will not join this week and will continue next week“*

Tabelle 5.4 Ausfälle von Sitzungen, die durch Entwickler2 verursacht wurden

Diese Absage offenbarte mir weitere Erkenntnisse darüber, wie die Entwickler mit Ausfällen umgingen. Wieder sprachen sie nicht darüber, dass einer der Entwickler sich verspätete bzw. ganz ausfiel.
Ich konnte mir folgende zwei Erklärungen dafür vorstellen: (1) die Entwickler hielten die Rolle des dritten Entwicklers in den Sitzungen für nicht wichtig und/oder (2) Konflikte wurden entweder ohne meine Anwesenheit oder gar nicht angesprochen.
Für die erste Erklärung sprach die Erkenntnis aus Abschnitt 5.2.1, dass Entwickler2 bereits seine Rolle als zweiter Entwickler in der Sitzung als nicht so relevant erachtete. Wahrscheinlich kam ihm dann die Rolle eines dritten Entwicklers überflüssig vor. Die Entwickler hatten offenbar au-

ßerdem ohne sich untereinander austauschen zu müssen alle die gleiche Einstellung, dass Sitzungen mit drei Entwicklern keine weiteren Vorteile erbringen würden bzw. zu viele Ressourcen in Anspruch nahmen. Ich kam zu dieser Einsicht aus mehreren Gründen: (1) hatte Entwickler1 mir in einem der späteren Einzelgespräche mit genau dieser Begründung mitgeteilt, dass er nicht plane den dritten Entwickler (Entwickler3) an den bereits geplanten Sitzungen zu beteiligen. (2) hatte auch Entwickler2 in der abschließenden Erhebung angedeutet, dass er sich als dritter Entwickler nicht in die Sitzung einbringen konnte. (3) haben mir sowohl Entwickler1 als auch Entwickler2 in der abschließenden Erhebung versichert, dass sie sich über ihre Erfahrungen in Sitzungen zu dritt nicht ausgetauscht hatten.
Die zweite Erklärung, dass die Entwickler grundsätzlich eher konfliktscheu waren, kam folgendermaßen auf. Die Sitzungsinhalte machten es nicht erforderlich, dass Entwickler2 dabei sein musste. Aber hätte es nicht selbstverständlich sein müssen, dass Entwickler1 Entwickler3 nach dessen Tischnachbarn fragte, wenn dieser unangekündigt nicht erschien? Stattdessen vergingen mehr als 20 Minuten der Sitzung bis Entwickler2 sich von selbst meldete, um seine Abwesenheit zu erwähnen, ohne sie aber näher zu erklären. Auch diesmal wurde weder von Entwickler1 noch von Entwickler3 näher hinterfragt bzw. eine Erklärung für die Absage gefordert. Dies war ein deutliches Indiz dafür, dass Geschehnisse, die Konfliktpotential hatten, von den Entwicklern nicht offen ausgesprochen wurden.
Es hatte auf diese Sitzung keine weiteren Auswirkungen. Die Sitzung konnte wie geplant durchgeführt werden, da die Sitzungsinhalte vorrangig mit Entwickler3 geplant waren und Entwickler2 für die Durchführung nicht essentieller Beteiligter war.
Für den Etablierungsprozess hatte es die Auswirkung, dass es die erste Sitzung mit drei Entwicklern werden sollte.

Busy II/No-Show III und Out-of-Office/No-Show IV

Auch die nächsten beiden abgesagten Sitzungen von Entwickler3 verstärkten den Verdacht von vorher, dass die Entwickler Sitzungen zu dritt weniger Priorität einräumten als andere Aufgaben zu erledigen. Wieder blieb das Nichterscheinen unkommentiert.

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
11, 12	Zu Sitzungsbeginn per Email	*"As I could not attend last session too, and this week also I am busy in some other tasks; that's why I would like to skip today's session. Only [Entwickler2] would be joining from here. Is this fine?"*

Tabelle 5.5 Ausfälle von Sitzungen, die durch Entwickler3 verursacht wurden

Obwohl Entwickler3 seine Beteiligung erst gar nicht (Out-of-Office wegen Krankheit) und beim zweiten Mal nur kurzfristig und per Email (Busy wegen dringenderer Aufgaben) mitteilte, hatte dies keine weiteren Auswirkungen auf den Sitzungsablauf. Die Sitzungen konnten ohne ihn abgehalten werden, da Entwickler2 anwesend war und er hauptsächlich in die Entwicklung dieses Features involviert war.
Die Auswirkungen der verschiedenen Formen des Nichterscheinens kann man folgendermaßen zusammenfassen. Das Fehlen des dritten Entwicklers hatte keine Auswirkungen auf die Fertigstellung der Sitzungsaufgaben. Jedoch konnte nicht genauer untersucht werden in welcher Form ein dritter Entwickler möglicherweise die Arbeit seiner Kollegen im Entwicklungsprozess unterstützt hätte. Es wurde nicht darüber gesprochen, warum Entwickler3 nicht erschien, aber damit hatte er endgültig den Anschluss an den Etablierungsprozess verloren. Die 12. Sitzung war der letzte Versuch ihn in die DPP-Sitzungen weiter einzubeziehen.

Sitzungs-nummer	Abgesagt durch	Grund	Zeitpunkt und Art der Absage	Verschoben um	Probleme / Auswirkungen
2	Entwickler3	nicht genannt (no show)	keine	nicht verschoben	fehlende Prozess-einführung
7	Entwickler2	nicht genannt (no-show)	in der Sitzung	nicht verschoben	fehlender Nutzen von DPP zu dritt, konfliktscheu
11	Entwickler3	Krankheit (no-show)	in der Sitzung	nicht verschoben	Distanzierung vom Etablie-rungsprozess
12	Entwickler3	dringendere Aufgaben (no-show)	zu Sitzungsbeginn	nicht verschoben	Distanzierung vom Etablie-rungsprozess

Tabelle 5.6 Zusammenfassung der Ausfälle von Entwickler2 oder Entwickler3

5.2.2.3 Ausfall durch Entwickler2 bzw. Entwickler3

Busy III

In den vorherigen Analysen hat sich herausgestellt, dass ein Ausfall von Entwickler2 oder Entwickler3 nicht dazu führte, dass Sitzungen ausfielen. Dies änderte sich jedoch, wenn beide Entwickler nicht zur Verfügung standen. Die Ursachen, die dazu führten werden im folgenden Abschnitt erörtert.
Bei den vorherigen Ausfällen spielte der Zeitpunkt der Absage eine eher kleine Rolle. Dies war nicht immer der Fall.

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
9	4 h vorher per Email	*"I apologize to inform you so late but for the next couple of days I have got some urgent tasks to finish; so I would like to propose to postpone today's session on Tuesday. Please let me know if this is fine for both of you?"*

Tabelle 5.7 Ausfälle von Sitzungen, die durch Entwickler3 verursacht wurden

Die Absage erfolgte vier Stunden vor der geplanten Sitzung per Email. Dies war zwar relativ kurzfristig, aber es wäre evtl. genug Zeit gewesen, um mit wenig Aufwand neu zu planen und in der verbleibenden Zeit eine alternative Sitzung mit Entwickler2 anzusetzen. Aber Entwickler1 hatte bis zwei Stunden vorher darauf noch nicht reagiert. Aus diesem Grund schrieb Entwickler3 eine weitere Email: *"Please let me know if you both are fine with this session postponing?"*.
Der Grund dafür, dass die Absage erst zwei Stunden vor der Sitzung bemerkt wurde, lässt sich darauf zurückführen, dass wir uns in verschiedenen Zeitzonen befanden. Durch den Zeitunterschied von 3,5 Stunden zwischen Entwickler3 und uns war es zum Zeitpunkt der Absage 6:30 Uhr CET.
Unabhängig davon war nur an wenigen Stellen wahrnehmbar, dass das DPP-Projekt auf verschiedenen Kontinenten und in unterschiedlichen Zeitzonen durchgeführt wurde wie z.B. der Terminplanung.
Eine Alternative zum Ausfall der Sitzung wäre gewesen, die Sitzung stattdessen mit anderen Inhalten und Entwickler2 durchzuführen. Da die Sitzung aber erst so kurzfristig vorher absagt wurde, hatte Entwickler1 nicht mehr die Möglichkeit neu zu planen, sodass auch diese Sitzung verschoben werden musste. Eine weitere Auswirkung auf den Etablierungsprozess war, dass die für diese Woche geplante Reflexion zu Sitzung 8 und 9 dadurch ebenfalls entfiel und aufgrund des Zeitabstandes zum neuen Termin auch nicht mehr nachgeholt werden konnte.

Zum ersten Mal hing es nicht von Entwickler1 ab, dass eine Sitzung ausfiel. Wie die nächste Analyse zeigt, gab es weitere Ausfälle, bei denen andere Personen über den Ausfall von Sitzungen entschieden.

Busy IV

In der 16. Sitzung teilte Entwickler1 mit, dass ihm vom Abteilungsleiter in Bangalore mitgeteilt wurde, dass Sitzung 17 um 4 Wochen verschoben werden müsste, weil Entwickler2 und Entwickler3 für die nächsten 3 Wochen mit anderen Aufgaben beschäftigt sein würden.
Da es für die aktuelle Aufgabe bereits eine Fertigstellungsfrist für Ende des Monats gab, teilte Entwickler1 in der Sitzung mit, dass er das Feature alleine fertigstellen würde.
Zusammenfassend kann man also festhalten, dass sowohl zu kurzfristige Absagen (aufgrund verschiedener Zeitzonen), die manchmal die Planung einer Alternative unmöglich machten, als auch das Einwirken von Dritten zu Sitzungsausfällen führten.

Sitzungs-nummer	Abgesagt durch	Grund	Zeitpunkt und Art der Absage	Verschoben um	Probleme / Auswirkungen
9	Entwickler3	dringendere Aufgaben	4 h vorher per Email	4 Tage	Kurzfristige Absage (Zeitzonen), keine Alternative
17	Entwickler1	dringendere Aufgaben	1 Woche vorher in der Sitzung	3 Wochen	Einfluss von außen

Tabelle 5.8 Zusammenfassung der Ausfälle von Entwickler2 oder Entwickler3

5.2.2.4 Ausfall wegen fehlender Betreuung

Es ist bereits ausführlich erörtert worden, dass ohne die Entwickler keine DPP-Sitzungen stattfinden konnten. Dagegen sollte die Betreuung in den DPP-Sitzungen eigentlich nur eine unterstützende Maßnahme sein. Es stellte sich jedoch heraus, dass sie ein essentieller Bestandteil für das Zustandekommen von Sitzungen darstellte.

Fehlende Voraussetzung I

Die 2. Sitzung konnte nicht wie geplant als erste Sitzung für den produktiven Einsatz von DPP genutzt werden, weil die Freigabe für die Inhalte der Arbeit von Teles nach der Unterzeichnung der Vereinbarung zur Verschwiegenheit (NDA) von ihnen noch nicht bewilligt worden war.

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
2	1 Tag vorher per Email	*"As you may have heard already we have to wait until next week with our code review because of the missing NDA. We have decided to hold a short session on Wednesday 10 am anyway in order to give [Entwickler2] and [Entwickler3] a short introduction into Saros with a sample project and to test performance and delay over a such long distance. After that, you will have some time to introduce your questionnaire and to give a view about your role in this dpp trial if you want ;-)"*

Tabelle 5.9 Ausfall des produktiven Einsatzes von DPP wegen fehlender Freigabe (NDA) für die Betreuung

Diese Sitzung zeigte besonders deutlich, welche Fragen zur Erörterung von Ausfällen relevant waren. Beispielsweise zeigte es, dass generell Entwickler1 über das Zustandekommen bzw. Sitzungsausfälle und Sitzungsinhalte entschied. In diesem Fall hatte er den produktiven Teil der Sitzung ausfallen lassen, da keine Betreuung zur Verfügung stand. Er hatte sich des Weiteren eine Alternative für den geplanten Sitzungstermin überlegt und dies einen Tag vorher per Email mitgeteilt, um die anderen Beteiligten darauf vorzubereiten.
In diesem Fall konnte der Sitzungsausfall durch eine alternative Planung wenigstens teilweise kompensiert werden.

Out-of-Office II

Es wurde bereits vorher erwähnt, dass Krankheitsbedingte Ausfälle (auch ohne Absage) der Entwickler vorgekommen waren, ohne dass sich dies auf den Sitzungsverlauf ausgewirkt hatte. Für den Ausfall der Betreuung galt dies offensichtlich nicht.

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
4	1 Tag vorher per Email	*"I am sorry, but this time I have to cancel the session. Because of health problems I will not be able to be in the office tomorrow. It would be important to me to attend the next session, but if you have urgent things to work on, please feel free to do the session tomorrow without me."*

Tabelle 5.10 Absage der Betreuung für die 4. Sitzung

Insgesamt fiel Sitzung 4 schon zum zweiten Mal aus (siehe Tabelle 5.14). Die technischen Voraussetzungen waren vorhanden, sodass die Entwickler nicht auf die Betreuung angewiesen waren. Doch die Entwickler hielten es anscheinend nicht für nötig eine DPP-Sitzung durchzuführen. Daher ließen sie den Termin ohne weitere Reaktion ausfallen. Es stellte sich an dieser Stelle die Frage, ob dieses Verhalten etwas über die Sicht der Entwickler auf DPP zeigte. Es gab darauf mehrere Antwortmöglichkeiten, z.B. (1) die Entwickler hielten nicht viel von DPP und ließen die Sitzungen, wenn möglich, ausfallen oder (2) sie führten mehr Sitzungen als erforderlich durch, weil das Projekt Teil eines Forschungsprozesses war.
Man könnte daraus ebenfalls den Schluss ziehen, dass die Entwickler es mit dem Vorankommen ihres Projektes nicht eilig hatten. Normalerweise werden Fristen in Projekten eher zu kurz geplant und daher herrscht ständig Termindruck. Es stellte sich jedoch später in einem Gespräch mit Entwickler1 über die Planung zum Projekt heraus, dass in den Sitzungen teilweise für ein späteres Release vorgearbeitet wurde und das Projekt trotz der Sitzungsausfälle noch gut in der Zeit lag. Selbst wenn Entwickler1 versicherte, dass die Aufgaben des DPP-Projektes die gleiche Priorität wie alle anderen seiner Aufgaben hatten, lässt sich nicht ausschließen, dass fehlender Termindruck zu mehr Ausfällen geführt hatte.
Obwohl die Entwickler sich nicht kritisch über die mehrfachen Sitzungsausfälle äußerten, war eine der Auswirkungen Diskontinuität. Es konnte bei den wenigen Sitzungen (höchstens eine pro Woche) keine Routine im Umgang mit der technischen Infrastruktur entstehen. Die Arbeit blieb liegen (außerhalb der Sitzungen wurde nicht an den Aufgaben gearbeitet) und Ziele bzw. Probleme gerieten in Vergessenheit. In einem der Fragebogen antwortete Entwickler1 auf die Frage nach Herausforderungen in der nächsten Sitzung wie folgt: *„Long time since last session and changing requirements. We have to redefine tasks."*
Die Entwickler fingen praktisch nach einer längeren Pause wieder von vorne an Routine, z.B. bezogen auf Saros-Funktionalitäten, aufzubauen. Das folgende Beispiel beschreibt diesen Aspekt und leitet gleichzeitig auf die folgenden Beobachtungen aus dem Abschnitt zu technischen Ausfällen über. In Sitzung 4 konnte nicht mit Saros gearbeitet werden, weil Entwickler1 Probleme mit dem Update der Software hatte und es auch in der Sitzung nicht mehr schaffte. Da Entwickler2 sich zehn Sitzungen später nicht mehr daran erinnern konnte, dass Updates vor einer Sitzung kritisch sein können, fiel aus demselben Grund auch Sitzung 14 aus. Entwickler1 kommentierte dies

in der Reflexion zur 14. Sitzung wie folgt:*„We have to keep in mind if there is an update that we do it in advance to avoid something like that today.“* Entwickler2 sagte dazu: *„Every time it worked so I never thought it would be a problem”*

Sitzungs-nummer	Abgesagt durch	Grund	Zeitpunkt und Art der Absage	Verschoben um	Probleme / Auswirkungen
2	Entwickler1	fehlende Vorbereitung	1 Tag vorher per Email	nicht verschoben	niedrige Priorität von DPP
4	Beobachterin	Krankheit	1 Tag vorher per Email	1 Woche	Diskontinuität

Tabelle 5.11 Zusammenfassung der Sitzungsausfälle wegen fehlender Betreuung

5.2.2.5 Ausfälle wegen fehlender Voraussetzungen und technischer Probleme

Wie bereits in Abschnitt 2.2 erwähnt, hat DPP eine Schwachstelle, weil für das Zustandekommen einer Sitzung die erforderliche technische Infrastruktur bei *allen* Beteiligten funktionieren muss. Es gab mehrere Sitzungsausfälle aufgrund von technischen Problemen. Wie die weitere Analyse zeigen wird, spielten neben technischen Problemen jedoch auch immer noch andere Faktoren eine Rolle.
Insgesamt gab es während des Etablierungsprozesses sieben Saros-Releases, entsprechend dem dreiwöchigen Veröffentlichungszyklus des Saros-Projekts. Da in jedem dieser Releases so entscheidende Änderungen an Saros vorgenommen wurden, dass keine Rückwärtskompatibilität zu älteren Versionen bestand, bedeutete die Veröffentlichung einer neuen Saros-Version, dass zwingend alle oder keiner der Sitzungsteilnehmer ihre Version zu aktualisieren hatten. Wie im Beispiel am Ende des letzten Abschnitts beschrieben führten zwei verschiedenen Releases (9.5.8.r1275 und 9.7.31.r1565) zu Problemen, da die Entwickler unterschiedliche Saros-Versionen installiert hatten.
In Sitzung 4 hatte Entwickler1 das Saros-Update und in Sitzung 14 hatte Entwickler2 das Saros-Update nicht durchgeführt. In beiden Fällen war den Entwicklern mindestens drei Tage vor der Sitzung empfohlen worden das Update vorzeitig durchzuführen und entsprechende Rückmeldung über den Verlauf zu geben, um die nötigen technischen Voraussetzungen für die Sitzung zu schaffen.
In den beiden folgenden Abschnitten wird im Detail erörtert, wie es zu den Ausfällen kommen konnte und welche Auswirkungen sie hatten.

Technische Probleme und fehlende Vorbereitung I

Für den Ausfall in der 4. Sitzung gab es verschiedene Gründe. Erst ihr Zusammenwirken verursachte den Ausfall. Der erste Grund war, dass Entwickler1 sich nicht auf die Sitzung vorbereitet hatte, weil er das empfohlene Update nicht vor der Sitzung durchgeführt hatte. Zweitens hatten die Entwickler sich nicht gut aufeinander abgestimmt. Denn es wäre auch möglich gewesen, dass beide vorerst mit einer gleichen aber älteren Version weiterarbeiten. Drittens hatten andere technische Probleme (Updates in Eclipse, die das Saros-Update verhinderten) Entwickler1 innerhalb der Sitzung daran gehindert das Saros-Update nachzuholen.

Eigentlich ist es nichts Ungewöhnliches, wenn beim Einsatz einer neuen Technologie am Anfang des Projektes eine Sitzung wegen technischer Probleme ausfällt. Doch wenn man den Kontext näher betrachtet, kann man besser verstehen, welche Auswirkungen sich aus dem Ausfall ergaben. Die Anfangsphase war für den Fortbestand der DPP-Sitzungen bei Teles am kritischsten. Es war keineswegs selbstverständlich, dass weiterhin Sitzungen stattfinden würden. Saros war noch eine α-Version und der Einsatz von DPP hing davon ab, ob sich die Sitzungen als technisch machbar und für den Entwicklungsprozess als hilfreich erweisen würden. Es hatte sechs Wochen nach dem Projektstart erst eine produktive Sitzung stattgefunden und dies war die erste Sitzung nach weite-

ren 3 Wochen von ausgefallenen Sitzungen (siehe Tabelle 5.14). Daher war es umso wichtiger für das Fortbestehen des DPP-Projektes, dass endlich wieder eine produktive Sitzung stattfinden konnte.
Mit ein wenig Vorbereitung, in Form eines Updates vor der Sitzung, hätten die Probleme wahrscheinlich früher erkannt und noch vor der Sitzung behoben werden können. Das Problem war, dass die Entwickler bis zuletzt nicht davon überzeugt werden konnten, sich besser auf die Sitzungen vorzubereiten.
Die Entwickler äußerten sich nicht direkt über den Sitzungsausfall, doch war zu beobachten, dass ihre Motivation für DPP sank und die Erwartungen für technische Probleme stiegen. Dies zeigte sich durch die Angaben der Entwickler in den Fragebogen vor der Sitzung. Abbildung 2 zeigt einen Auszug der Priorität „keine technischen Probleme treten auf oder stören die Sitzung“ von Entwickler1 und Entwickler2.

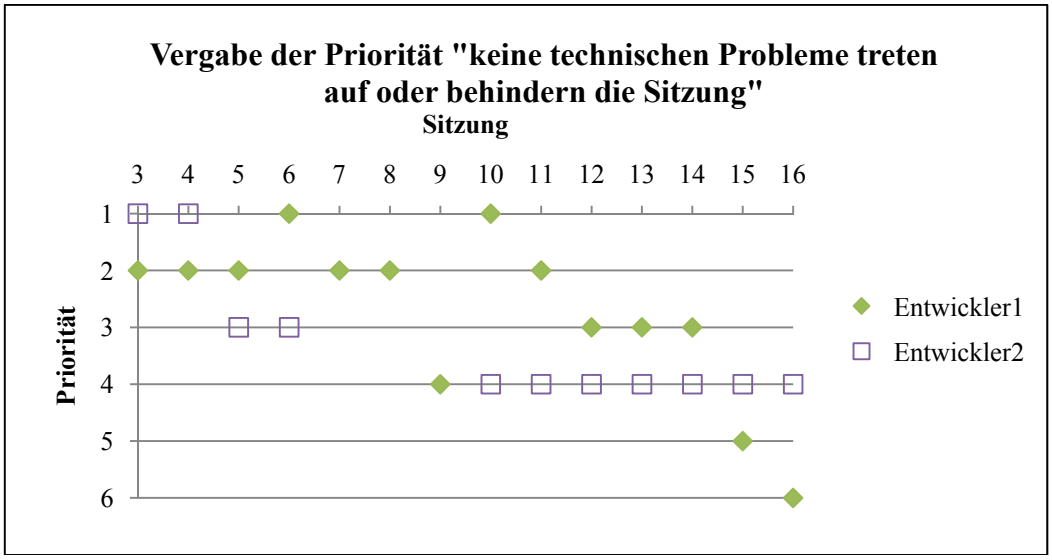


Abbildung 5.2 Angaben der Entwickler im Fragebogen US über die Priorität dafür, dass keine technischen Probleme die Sitzung behindern (US-kurz Frage 7, 1= sehr wichtig bis 6 = am wenigsten wichtig)

Es zeigte sich, dass Entwickler1 in den ersten acht Sitzung im Fragebogen vor der Sitzung stets angab, dass es für den Erfolg der Sitzung wichtig (in Sitzung 6 sogar sehr wichtig) war, dass keine technischen Probleme auftraten. Auch bei Entwickler2 war zu beobachten, dass er in den ersten Sitzungen technischen Problemen eine hohe Relevanz für den Erfolg der Sitzung beimaß. In späteren Sitzungen traten weniger technische Probleme auf, dementsprechend sank auch die Priorität der Entwickler im Fragebogen.
Auf die offene Frage, welche Probleme sie für die nächste Sitzung erwarteten, gaben sie stets und ausschließlich technische Probleme an (siehe US-kurz Frage 6).
Damit zeigte sich die Notwendigkeit, die Sitzungen soweit wie möglich von technischen Problemen zu befreien. Zum einen damit die Entwickler ihre Arbeit erledigen konnten und die Sitzungen als produktiv erlebten und somit motiviert blieben. Zum anderen, damit sie sich darauf einlassen würden auch anderen Aspekten der Sitzung (Inhaltliches oder Zwischenmenschliches) mehr Aufmerksamkeit zu schenken.

Technische Probleme und fehlende Vorbereitung II

Wie in der 4. Sitzung stellte sich auch in der 14. Sitzung heraus, dass Saros wegen verschiedener Versionen nicht verwendet werden konnte (siehe Abschnitt 4.7.5).
Wieder gab es verschiedene Ursachen, dass Saros nicht verwendet werden konnte. Wie vorher erörtert, hätten eine bessere Abstimmung zwischen den Entwicklern oder eine frühere Vorbereitung weitere Probleme vermieden. Hinzu kam jedoch, dass Entwickler2 eine falsche Angabe gemacht hatte. Er hatte Sitzungszeit verschwendet, um ein technisches Problem bzw. seine zu kurz-

fristige Vorbereitung zu vertuschen. Dies hätte durch eine zusätzliche Desktop Sharing-Verbindung zu ihm evtl. verhindert werden können.
Dieser Vorfall hatte zwei Auswirkungen, die besonders erwähnenswert sind. Erstens konnte die Sitzung, anders als Sitzung 4, auch ohne Saros von den Entwicklern wenigstens teilweise für produktive Arbeit genutzt werden. Zweitens ergab sich daraus für den Etablierungsprozess eine wichtige Erkenntnis. Das Problem war am Ende der Sitzung gelöst, aber es hatte die gesamte Sitzungszeit und zusätzlich noch die Arbeitszeit von vier weiteren Saros-Entwicklern erfordert. Als Konsequenz aus den Geschehnissen in Sitzung 14 beschloss das Saros-Team eine simple, aber nötige, interne Vorgehensweise festzulegen, für den Fall, dass Sitzungen wegen fehlender Vorbereitung oder technischer Probleme nicht stattfinden konnten (siehe 5.3.6 Neuerungen durch Teles). Damit sollte der Ablauf optimiert und so wenig Zeit wie möglich verschwendet werden.

Technische Probleme III

Ein weiterer Vorfall, der im weitesten Sinne mit der technischen Infrastruktur zusammenhing, war der Ausfall der 14. Sitzung, weil das Notebook von Entwickler1 nicht einsatzbereit war.

Abgesagte Sitzung	Zeitpunkt und Art der Absage	Erklärung
14	3 h vorher per Email	*"Since my notebook is broken I am not able to participate the session today. We will continue next week as usual."*

Tabelle 5.12 Absage der 14. Sitzung von Entwickler1 wegen technischer Probleme

Da Sitzungen ohne Entwickler1 nicht stattfinden konnten, und er anscheinend nicht damit rechnete bis zur Sitzung Ersatz für sein Notebook inklusive aller nötigen Softwareinstallationen (siehe Abschnitt 4.3) zur Verfügung zu haben, musste auch diese Sitzung verschoben werden. Insgesamt fiel Sitzung 14 damit drei Mal aus (siehe Tabelle 5.14). Die nächste Sitzung fand erst vier Wochen später als geplant statt.

Sitzungs-nummer	Abgesagt durch	Grund	Zeitpunkt und Art der Absage	Verschoben um	Probleme / Auswirkungen
4	Entwickler1	technische Probleme	in der Sitzung	1 Woche	fehlende Vorbereitung, sinkende Motivation
14	Entwickler1	technische Probleme	3 h vorher per Email	1 Woche	fehlende Vorbereitung, Falschangabe
14	Entwickler2	technische Probleme	in der Sitzung	1 Woche	fehlende Vorbereitung, Hardwareausfall

Tabelle 5.13 Zusammenfassung der Sitzungsausfälle wegen fehlender Voraussetzungen oder technischer Probleme

5.2.2.6 Zusammenfassung

Tabelle 5.14 fasst alle vorher beschriebenen Vorfälle zusammen. Anhand sich wiederholender Sitzungsnummern kann man erkennen, wie oft eine Sitzung verschoben werden musste. Außerdem werden in der Tabelle noch die Gründe, Details zur Absage und Stichpunkte über die Erkenntnisse aufgeführt.

Sitzungs-nummer	Abgesagt durch	Grund	Zeitpunkt und Art der Absage	Verschoben um	Probleme / Auswirkungen
2	Entwickler1	fehlende Vorbereitung	1 Tag vorher per Email	nicht verschoben	niedrige Priorität von DPP
2	Entwickler3	nicht genannt (no-show)	keine	nicht verschoben	fehlende Prozess-einführung
4	Entwickler1	dringendere Aufgabe	1 Tag vorher per Email	1 Woche	unbeständige Pla-nung, Projektverzögerung
4	Betreuerin	Krankheit	1 Tag vorher per Email	1 Woche	Diskontinuität
4	Entwickler1	technische Probleme	in der Sitzung	1 Woche	fehlende Vorbereit-tung, sinkende Motivation
7	Entwickler2	nicht genannt (no-show)	in der Sitzung	nicht verschoben	fehlender Nutzen von DPP zu dritt, konfliktscheu
9	Entwickler1	Anderes Mee-ting	1 Tag vorher in der Reflexion	1 Tag	unbeständige Pla-nung, Projektverzögerung
9	Entwickler3	Dringendere Aufgaben	4 h vorher per Email	4 Tage	Kurzfristige Absage (Zeitzonen), keine Alternative
11	Entwickler3	Krankheit (no-show)	in der Sitzung	nicht verschoben	Distanzierung vom Etablierungsprozess
12	Entwickler3	dringendere Aufgaben (no-show)	zu Sitzungsbe-ginn	nicht verschoben	Distanzierung vom Etablierungsprozess
13	Entwickler1	dringendere Aufgaben	1 h vorher per Email	1 Woche	unbeständige Pla-nung, Projektverzögerung
14	Entwickler 1	spontaner Urlaub	1 Woche vorher in der Sitzung	1 Woche	kurzfristige Pla-nung, Projektverzö-gerung
14	Entwickler1	technische Probleme	3 h vorher per Email	1 Woche	fehlende Vorbereit-tung, Falschangabe
14	Entwickler2	technische Probleme	in der Sitzung	1 Woche	fehlende Vorbereit-tung, Hardwareaus-fall
17	Entwickler1	dringendere Aufgaben	1 Woche vorher in der Sitzung	3 Wochen	Einfluss von außen

Tabelle 5.14 Übersicht über Ausfälle von Sitzungen oder ausgefallene Entwickler

Aus der Analyse lässt sich insbesondere erkennen, dass bei Absagen, die durch äußere Einflüsse oder andere Aufgaben verursacht wurden, die Rollenverteilung im DPP-Projekt manchen Beteiligten erlaubte von Sitzungen fernzubleiben, ohne dass sich dies auf den Sitzungsverlauf auswirkte. Wiederum andere Personen waren essentielle Beteiligte an einer Sitzung, sodass durch ihre Absagen Ausfälle und Verzögerungen verursacht wurden. Verschiedene Zeitzonen führten dazu, dass Absagen zu kurzfristig ankamen und keine Alternative mehr geplant werden konnte.
Es zeigte sich durch das Nichterscheinen der Entwickler, dass sie generell aus Sitzungen zu dritt keinen Nutzen ziehen konnten. Sie waren jedoch zu konfliktscheu, um offen darüber zu reden.
Des Weiteren zeigte sich, dass Ausfälle, die durch technische Probleme hervorgerufen wurden, teilweise durch flexible Vorgehensweise (und alternative Inhalte) kompensiert werden konnten.
Eine unerwartet positive Folge der Ausfälle war, dass in der Zwischenzeit gravierende Probleme wie z.B. Probleme mit der technischen Infrastruktur ausführlich behandelt oder sogar behoben werden konnten. Weiterhin ermöglichte die freie Zeit die durch Sitzungsausfälle zur Verfügung stand, die vorhandenen Daten genauer zu analysieren.

Probleme mit der technischen Infrastruktur zeigten sich vor allem als Grund für einen Ausfall, wenn zusätzlich noch andere Faktoren wie z.B. fehlende Vorbereitung oder fehlende Abstimmung zwischen den Beteiligten vorlag. Ausfälle, die durch technische Probleme verursacht wurden führten dazu, dass die Motivation der Entwickler für die DPP-Sitzungen sank und die Erwartung für technische Probleme sich erhöhte. In späteren Phasen, in denen technische Probleme weniger dominant auftraten, zeigte sich, dass diese in der Anfangsphase dafür gesorgt hatten, dass andere (nicht technische) Ereignisse des DPP-Projektes dadurch überschattet wurden.
Auch alltägliche Gründe, wie spontaner Urlaub und Krankheit führten zu Sitzungsausfällen. Nach längeren Pausen, die durch Ausfälle verursacht wurden, zeigten sich deutliche Zeichen von Diskontinuität im Entwicklungs- sowie im Etablierungsprozess, da z.B. keine Routine mehr im Umgang mit der technischen Infrastruktur vorhanden war und Reflexionen entfielen. Für den Etablierungsprozess war der Ausfall von Sitzungen am Anfang besonders gravierend, da die Existenz des DPP-Projektes davon abhing, dass die Entwickler das Projekt als produktiv empfanden. Schließlich kam es wegen Ausfällen innerhalb der ersten sechs Wochen des DPP-Projektes nur zu zwei produktiven Sitzungen. Wegen mehrfachen Ausfällen, konnten Erweiterungen des Etablierungsprozesses, z.B. weitere Sitzungstypen und Sitzungen zu dritt, kaum untersucht werden.
Die Ausfälle verringerten die Anzahl der möglichen Sitzungen im Etablierungsprozess erheblich. Die Fertigstellung der Entwicklungsaufgaben war jedoch noch innerhalb der Deadline möglich.

5.2.3 Verspätungen

Im Verlauf des DPP-Projektes kam es zu mehreren Verspätungen, weshalb einige Sitzungen nur verkürzt stattfanden. Abbildung 5.4 zeigt eine Übersicht über das Verhältnis von Verspätungen, weiteren Verzögerungen und der restlichen Sitzungszeit für alle Sitzungen. Im folgenden Abschnitt sollen die Gründe, Wechselwirkungen und Bedeutungen von Verspätungen für eine Prozesseinführung von DPP erläutert werden.

Es gab verschiedene Arten von Verspätungen:

- Verspätungen zu Sitzungen: Ein Entwickler verspätete sich zu Sitzungen.
- Verspätungen zu Reflexionen: Entwickler kamen zu Reflexionen zu spät.
- Verspätungen durch Reflexionen: Reflexionen vor Sitzungen verschoben den geplanten Beginn der folgenden Sitzungen.

Für die Analyse von Verspätungen waren folgende Fragen interessant:

- Wer verspätete sich?
- Wozu verspätete er sich?
- Wie wurde damit umgegangen?

Im folgenden Abschnitt wird im Detail darauf eingegangen, wie es zu Verspätungen kam und welche Auswirkungen sie auf den DPP-Prozess hatten. Nacheinander werden die Phänomene zu verschiedenen Arten von Verspätungen, d.h. zu Sitzungen, zu Reflexionen und durch Reflexionen, beschrieben. Die Ergebnisse werden jeweils anhand von Stichworten in einer Tabelle zusammengefasst. In der Zusammenfassung dieses Abschnitts befindet sich eine Übersicht über alle Verspätungen (siehe Tabelle 5.21).

5.2.3.1 Verspätungen von Entwickler2 zu Sitzungen

Entwickler2 verspätete sich über 15 Minuten zu mehreren Sitzungen während des Etablierungsprozesses (siehe Tabelle 5.15).

Termin	Entwickler	Verspätung in Minuten	Grund
Sitzung 5	2	17	unbekannt
Sitzung 7	2	23	unbekannt
Sitzung 11	2	15	unbekannt
Sitzung 13	2	17	längere Mittagspause

Tabelle 5.15 Verspätungen von Entwickler2 zu Sitzungen

Grundsätzlich gab es bei Entwickler2 Widersprüche zwischen dem Verhalten (den Verspätungen) und den Angaben diesbezüglich im Fragebogen. Wie schon in Abschnitt 5.2.1 erläutert, hätte es ihm wichtig sein müssen, die gesamte Sitzungszeit zur Verfügung zu haben. In den meisten Fragebogen gab er an, dass es für den Erfolg der Sitzung wichtig war die Aufgabe pünktlich zu beenden (siehe Abbildung 5.3).

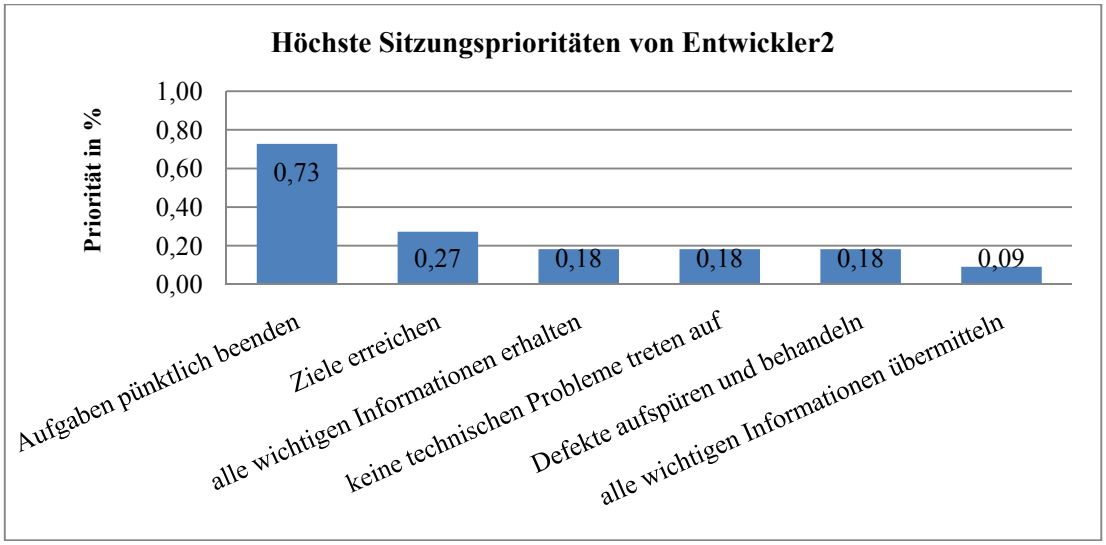


Abbildung 5.3 Angaben im Fragebogen US zu Frage 7 von Entwickler2 über die höchsten Prioritäten für den Erfolg der Sitzung (mit Mehrfachangaben)

Abbildung 5.3 zeigt, welche Faktoren für Entwickler2 die höchste Relevanz für den Erfolg von Sitzungen hatten. Entwickler2 gab in 73 % der Fragebogen „die Aufgabe pünktlich beenden“ als einen der wichtigsten Faktoren für den Erfolg der Sitzung an. Darunter waren auch die Fragebogen aus den Sitzungen 11 und 13.
Im Weiteren wird im Detail erörtert, wie es zu dem Widerspruch zwischen den Sitzungszielen und dem Verhalten von Entwickler2 kommen konnte. Die Verspätungen werden abhängig von den Sitzungstypen betrachtet. Zuerst wird erörtert wie es zu der Verspätung zum Review in Sitzung 7 kam. Danach werden die Verspätungen zu Pair-Developing-Sitzungen (5, 11 und 13) gemeinsam untersucht.

Verspätungen von Entwickler2 zu einem Review

Wie bereits zum Nichterscheinen zu Sitzungen im vorherigen Abschnitt erläutert hatte Entwickler1 Entwickler2 für das Review in der 7. Sitzung eingeplant (vgl. Abschnitt 5.2.2). Er hatte den beiden Entwicklern aus Bangalore sogar die gleiche Rolle zugedacht, obwohl der Code von Entwickler3 erzeugt worden war. Seine Angabe über die geplanten Aufgaben im Fragebogen lautete: *„Synchronizing the Projects. Code Review presented by [Entwickler3] and [Entwickler2] followed by a discussion.*“
Obwohl Entwickler1 angab, dass Entwickler2 und Entwickler3 für die Sitzung die gleichen Rollen hatten, handelte es sich schließlich wieder um eine Sitzung, bei der Entwickler2 nicht der Autor des Quellcodes war. Dieses Mal war er sogar nur der dritte beteiligte Entwickler. Folglich würde die Vermutung über sein Verständnis der Rollenverteilung und Verantwortlichkeiten wie sie bereits in Abschnitt 5.2.1 erläutert wurden, nahelegen, dass er seine eigene Rolle in dieser

Konstellation als noch überflüssiger als in DPP-Sitzungen zu zweit ansah und sich daher lieber mit anderen Aufgaben befasste. Dies würde erklären, warum er erst 23 Minuten nach Sitzungsbeginn Bescheid gab, dass er nicht teilnehmen würde. Das Ausbleiben einer Reaktion der anderen Entwickler hätte er als Bestätigung für sein Handeln auffassen können.
Zu diesem Zeitpunkt waren sich die Entwickler in Bezug auf die Rolle eines dritten Entwicklers noch nicht einig. Schließlich hatte Entiwckler1 Entwickler2 eine aktive Rolle zugedacht. Man hätte daher als Reaktion erwarten können, dass die anderen Entwickler die Verspätung oder den Ausfall von Entwickler2 auf irgendeine Weise kommentierten. Doch weder in der Sitzung noch in der Reflexion wurde von den Entwicklern zur Sprache gebracht, dass oder weswegen Entwickler2 nicht wie geplant zur Sitzung erschienen war. Dies und die spärliche Kommunikation innerhalb der Reflexionen verstärkten erneut den Eindruck, dass die Entwickler nicht in meiner Anwesenheit oder überhaupt nicht über Konflikte sprachen.

Verspätungen von Entwickler2 zu Pair-Developing-Sitzungen

Vermutlich sah sich Entwickler2 auch bei keiner der Pair-Developing-Sitzungen für den Erfolg der Sitzung verantwortlich. Dies wäre zumindest eine Erklärung dafür, warum er mehrere Male erheblich zu spät zu Sitzungen dieses Typs erschien (vgl. Tabelle 4.11 und Tabelle 5.15).
Bei keiner seiner Verspätungen machte er sich die Mühe eine Begründung anzugeben und die anderen Entwickler forderten auch keine von ihm. Dies machte es relativ schwierig die wahren Gründe für die Verspätungen zu identifizieren. Es war zu befürchten, dass sich hinter den Verspätungen ein größeres Problem verbarg. Aus diesem Grund war es schon während des Etablierungsprozesses relevant, sich trotzdem detailliertere Gedanken darüber zu machen.
Erste Vermutungen über die Gründe legten nahe, dass Entwickler2 im Allgemeinen zu Unpünktlichkeit neigte. Denn er kam öfter ein paar Minuten, d.h. drei bis fünf Minuten, zu spät. Eine weitere Vermutung war, dass sich der Tagesablauf in der neuen Arbeitsumgebung noch nicht reguliert hatte und sich dadurch ab und zu vorherige Aufgaben verzögerten. Eine weitere plausible Erklärung für die Verspätungen von Entwickler2 war, dass sie ein Zeichen dafür waren, dass er seine Grenzen und damit seine Position im neu gebildeten Team testen wollte.[16] Es war hingegen auch denkbar, dass die Sitzungszeiten sich mit den üblichen Mittagszeiten seiner Kollegen überschnitten und er auch an Sitzungstagen nicht auf die gemeinsamen Pausen verzichten wollte. Dies würde zumindest seinen einzigen Erklärungsversuch im letzten Gespräch darüber bestätigen. Aber darauf wird später noch genauer eingegangen.
Es drängte sich im Besonderen noch ein weiterer Verdacht als Ursache für die Verspätungen auf, der eine etwas komplexere Erklärung hat. An verschiedenen Stellen zeigte sich, dass Entwickler2 in Sitzungen vom Typ Pair-Developing (bei denen er nicht ursprünglicher Autor des Quellcodes war, also alle Sitzungen nach der dritten) so gut wie keine Anstrengungen unternahm, Verantwortung im DPP-Projekt zu übernehmen. Dies äußerte sich bspw. darin, dass er nur selten eine aktive Rolle in den Sitzungen übernahm. Folglich könnte die Ursache der Verspätungen auch mangelnde Motivation gewesen sein.
Dies lässt sich wie folgt herleiten. Die Anwesenheit des Observers als zweiter Entwickler bei DPP hat viele verschiedene Vorteile, z.B. Wissenstransfer und Qualitätsverbesserung. Dies ist zwar wissenschaftlich belegt (siehe Abschnitte 2.2 und 2.3.1), aber nicht jedem sofort ersichtlich. Daher kann nicht vorausgesetzt werden, dass Entwickler2 sich dessen bewusst war. Im Gegenteil ist eher zu vermuten, dass ihm dies nicht klar war. Erstens lässt es sich darauf zurückführen, dass er keine Erfahrungen mit PP und DPP hatte (siehe Tabelle 4.2) und dass er sich darunter auch anfangs nicht viel vorstellen konnte (siehe Tabelle 4.3). Zweitens hatte er im Gegensatz zu Entwickler1 Wissenstransfer nicht als explizites Projektziel angegeben (siehe Abschnitt 4.4.2). Drittens wurde zudem bereits in Abschnitt 5.2.1 erläutert, dass Entwickler2 ein Rollenverständnis hatte, das ihm (als nicht ursprünglicher Autor) keine besonderen Verantwortungen für die Zusammenarbeit bei DPP in Pair-Developing-Sitzungen zuwies. Als erstes Zwischenergebnis kann man also

[16] Nach Tuckman [Tuc65] entspricht dies der zweiten Phase (Storming) des 5-Phasenmodells des Teamentwicklungsprozesses.

festhalten, dass Entwickler2 sich vor dem Etablierungsprozess vermutlich nicht darüber bewusst war, dass auch der Observer eine wichtige Rolle bei DPP hat.
Nun stellte sich die Frage, warum es ihm nicht möglich war, während des Etablierungsprozesses andere (positivere) Erfahrungen bzgl. seiner Rolle bei DPP zu sammeln. Dies lässt sich durch ein in Abschnitt 2.3.1 bereits beschriebenes Phänomen, das Stephens und Rosenberg [SR03] als *Go Make Me a Cup of Tea - Syndrome* bezeichnen, erklären. Bei unterschiedlichen Erfahrungsstufen der beteiligten Entwickler (hier *Expert-Novice*) besteht das Risiko, dass der Experte dem Novizen durch sein Vorgehen keine Chance gibt sich zu beteiligen, weil er zu wenig Geduld hat oder dem Novizen keine Möglichkeiten zur Beteiligung einräumt. Obwohl er es nicht vorhatte, trug Entwickler1 also dazu bei, dass Entwickler2 sich nicht aktiv einbringen konnte, weil er kein klar definiertes Lernkonzept in den Sitzungen verfolgte und ihm manchmal auch die Geduld fehlte. Dies zeigte sich bspw. auch in Sitzung 5. Auf der einen Seite versuchte er spontan Entwickler2 die Driver-Rolle zu übergeben. Andererseits entzog er sie ihm wieder, als es ihm nicht schnell genug voranging (siehe chronologische Beschreibung der 5. Sitzung in Anhang F). Als zweites Zwischenergebnis kann man also festhalten, dass Entwickler2 in den Pair-Developing-Sitzungen die Erfahrung machte: (1) dass er als Novize bei DPP nicht viel beitragen konnte, (2) dass, gemessen an seinen Kriterien für eine erfolgreiche Sitzung (lauffähigen Code als Output zu erzeugen), seine Ziele ohne sein Zutun trotzdem erreicht werden konnten.
Wenn ihm also nicht bewusst war, dass er alleine durch seine Anwesenheit und das gelegentliche Hinterfragen der Aktivitäten von Entwickler1 schon einen großen Beitrag zu den Sitzungen leistete, dann ist es verständlich, dass er keinen wichtigen Grund für seine Teilnahme sah. Dementsprechend wird es ihm wahrscheinlich auch schwer gefallen sein sich überhaupt als zweiter Entwickler, in der Rolle des Observers und als Novize, zu motivieren an Pair-Developing-Sitzungen teilzunehmen. Damit sollte dargelegt sein, wie es zu der Vermutung über die Verbindung zwischen fehlender Motivation und Verspätungen von Entwickler2 kam.
Eine Erkenntnis daraus scheint zu sein, dass Entwickler, die bei DPP die niedrigere Erfahrungsstufe haben, z.B. Novizen, sich über die Vorteile von DPP bewusst sein müssen, weil sie dann erst ihre Rolle bei DPP verstehen und damit die nötige Motivation für eine aktive Teilnahme dabei entwickeln können.

Gemessen an den Projektzielen der Entwickler hatten die Verspätungen von Entwickler2 zu den Pair-Developing-Sitzungen keine negativen Auswirkungen, weil sowohl Wissenstransfer stattfand als auch lauffähiger Code erzeugt wurde. Die Sitzungsziele konnten meistens auch erreicht werden, z.B. weil eine kurze Synchronisationszeit die Verspätung kompensierte (siehe Sitzung 11), und falls nicht, dann machten die Entwickler andere Gründe dafür verantwortlich. So führte bspw. Entwickler1 nach der 13. Sitzung die Verzögerung im Plan nur auf aufgetretene Probleme zurück: *„we did some work ... which didn't cover the test."*
Betrachtet man jedoch den dadurch entstandenen Zeitverlust, hatten die Verspätungen sehr wohl einen negativen Einfluss auf die Bilanz der produktiven Sitzungszeit. Da es auch nach der dritten erheblichen Verspätung nicht den Anschein hatte, dass die Entwickler von selbst darauf zu sprechen kommen würden und um schlechten Gewohnheiten vorzubeugen, war es angebracht an dieser Stelle in den Etablierungsprozess einzugreifen.

Interventionen waren zudem notwendig, da die Entwickler grundsätzlich konfliktscheu waren. Dies zeigte sich wie folgt: Entwickler1 gab zwar an, dass ihm die Verspätungen von Entwickler2 nicht aufgefallen waren, weil sie zu gering waren. Dies würde auch erklären, warum er in den Fragebogen gekennzeichnet hatte, dass die Aufgaben nicht so schnell wie geplant ausgeführt werden konnten, dabei jedoch nie eine Verspätung als Begründung auftauchte. Gegen diese Begründung sprach jedoch, dass die Verspätungen insgesamt und besonders in der 11. Sitzung mehr als auffällig waren. Denn Entwickler1 war zusätzlich eine Viertelstunde früher zur Sitzung erschienen und musste daher nicht nur 15, sondern 30 Minuten bis Sitzungsbeginn warten.
Was den Verdacht, dass die Entwickler konfliktscheu sind, noch bestärkte, waren folgende drei Gründe: (1) Die Entwickler konnten ohne weiteres über das Ausbleiben des abwesenden Entwicklers3 sprechen, aber nicht über die Verspätung von Entwickler2, der anwesend war. (2) Sie sprachen nur sehr ungern in Reflexionen über Probleme und Erfahrungen aus den Sitzungen. (3) Ent-

wickler1 hatte auffällig lange damit gewartet ein Gespräch über Verspätungen mit Entwickler2 zu führen. Sie wollten vermutlich freiwillig weder über Verspätungen noch über andere bestehende Konflikte sprechen. Dies zeigte mir die Möglichkeit auf, dass es noch andere verborgene Konflikte geben konnte, die mir bis dahin nicht aufgefallen waren.
Ich nannte also Entwickler1 meine Vermutungen über die Motivation und das Rollenverständnis von Entwickler2 und schlug ihm dies als Thema für die nächste Reflexion vor. Einerseits wollte ich erfahren, ob es Entwickler1 aufgefallen war (und wenn ja, warum er nichts dazu gesagt hatte) und seine Meinung darüber erfahren. Andererseits wollte ich, dass er Entwickler2 darum bitten sollte Verspätungen zu vermeiden. Ich hoffte natürlich, dass ich dabei sein konnte, wenn Entwickler1 und Entwickler2 darüber sprachen, um meine zwei Vermutungen bzgl. der Konfliktscheue und über den Grund der Verspätungen zu überprüfen.
Es könnte nun die Frage naheliegen, warum ich mich nicht selbst an Entwickler2 wandte. Dies hatte drei Gründe: (1) am wichtigsten war, dass die Entwickler endlich anfingen direkt miteinander und ohne mich über nichttechnische Konflikte zu sprechen. (2) Ein Nebeneffekt war, dass ich mich durch weitere Ermahnungen nicht noch unbeliebter machte. Ich hatte bereits mehrmals auf die Beantwortung der Fragebogen bestehen oder verspätete Abgaben ermahnen müssen. Die Entwickler hatten ihrerseits deutlich gezeigt, dass der Mehraufwand der zusätzlichen Maßnahmen (Fragebogen, Reflexione, etc.) im Etablierungsprozess sie belastete. (3) Außerdem wollte ich vermeiden, von den Entwicklern noch mehr als Kontrollinstanz wahrgenommen zu werden. Es war schon schwierig genug sie in Reflexionen immer wieder auf Konflikte aufmerksam zu machen und all ihre Aktivitäten aufzuzeichnen, ohne dass sie sich dabei zu sehr überwacht fühlten.

Nach meinem Hinweis von vorher und der vierten auffälligen Verspätung von Entwickelr2 konnte Entwickler1 nicht mehr vermeiden es zu bemerken. Wie vorgeschlagen, sprach er dies in der 13. Reflexion an und bat Entwickler2 sich in Zukunft nicht mehr zu verspäten. Die gesamte Unterhaltung darüber dauerte genau 45 Sekunden und verlief wie folgt:

Entwickler 1: *„Is there any reason why you were late?* “
Entwickler 2: *“Actually, I went for lunch late, so I came late. It was raining so I got wet”*
Entwickler 1: *“[…] It is once a week every Wednesday, so I kindly ask you to be here at the right time”*
Entwickler 2: *“Yep”*

Die zwei positiven Auswirkungen waren, dass die Entwickler einmal direkt über einen Konflikt gesprochen hatten, und dass Entwickler2 danach nicht mehr zu spät kam.
Doch der wirkliche Grund für seine regelmäßigen Verspätungen wurde nicht geklärt. Ich war versucht ihn selbst noch einmal danach zu fragen, doch meine bisherige Erfahrung sagte mir, dass es unwahrscheinlich sein würde, eine offene Antwort von ihm zu erhalten. Entwickler2 hatte schon mehrfach angedeutet, dass er sich durch mich überwacht fühlte. In der 14. Reflexion sagte er bspw.: *„I think you are suspisious today“* auf die Frage, ob die Entwickler sich auch außerhalb der Sitzung über Anforderungen von Saros austauschen würden. Außerdem hatte ich schon mehrfach während des DPP-Projektes die Erfahrung gemacht, dass Entwickler2 sich durch diplomatische Antworten aus der Verantwortung stahl. Auf die Frage im ersten Fragebogen, ob er lieber im Team oder alleine an neuen Aufgaben arbeitete war seine Antwort bspw.: *„I don’t have any problem for anything“*. Daher machte ich keinen Versuch ihn selbst noch einmal danach zu fragen.

Da Entwickler2 sich in den folgenden drei Sitzungen nicht noch einmal verspätete, wurde dieser Punkt nicht mehr thematisiert. Es bleibt also offen, warum Entwickler2 sich wirklich zu den Sitzungen verspätete. Es ist auch gut möglich, dass es eine Kombination von mehreren Gründen war.

Termin	Entwickler	Verspätung in Minuten	Grund	Probleme/ Auswirkungen
Sitzung 5	2	17	unbekannt	konfliktscheu, falsches Rollenverständnis
Sitzung 7	2	23	unbekannt	Keine Rolle für den dritten Entwickler bei DPP
Sitzung 11	2	15	unbekannt	konfliktscheu, falsches Rollenverständnis
Sitzung 13	2	17	Mittagspause	konfliktscheu, falsches Rollenverständnis

Tabelle 5.16 Zusammenfassung von Verspätungen von Entwickler2 zu Sitzungen

5.2.3.2 Verspätungen zu Reflexionen

Es gab nicht nur Verspätungen zu Sitzungen sondern auch zu Reflexionen (siehe Tabelle 5.17).

Termin	Entwickler	Verspätung in Minuten	Grund
Reflexion 3	1, 2	15	Termin vergessen
Reflexion 10	2	10	unbekannt

Tabelle 5.17 Verspätungen der Entwickler zu Reflexionen

Bereits im Zusammenhang mit vorher erwähnten Verspätungen deutete sich an, dass die Entwickler alle Aktivitäten des Etablierungsprozesses (also auch die Reflexionen) als belastenden Mehraufwand empfanden. Die Reflexionen waren ein Teil des Etablierungsprozesses, sie dienten jedoch vorrangig dem Zweck, die Kommunikation zwischen den Entwicklern über Erfahrungen und Probleme in der Sitzung zu fördern. Erst im zweiten Schritt der Anpassung des Etablierungsprozesses gelang es die Bereitschaft der Entwickler für Reflexionen zu erhöhen. In welchem Zusammenhang dies mit Verspätungen steht, wird in den nächsten beiden Abschnitten (Verspätungen zu Reflexionen und Verspätungen durch Reflexionen) näher erläutert.

Verspätung zu Reflexionen I

Mir wurde bereits in der ersten Reflexion von beiden Entwicklern aufgezeigt, dass sie der Reflexion keinen hohen Stellenwert einräumten.
Die erste Reflexion startete mit 15 Minuten Verspätung. Ich wartete 10 Minuten vergeblich und schrieb ihnen dann eine Email. Daraufhin meldete sich zunächst Entwickler1, der dann per Chat Entwickler2 Bescheid gab. Daraufhin konnte die Reflexion doch noch stattfinden. Sie dauerte etwa 9 Minuten.

Als Grund für die Verspätung hatten die Entwickler angegeben, dass sie den Termin vergessen hatten. Ich erklärte mir die Verspätung damit, dass die Entwickler anfänglich ein falsches Bild der Rolle der Reflexion im DPP-Projekt hatten. Sie sahen es nicht als wichtigen Teil ihres Entwicklungsprozesses, sondern nur als eine weitere Form des Informationsaustauschs mit mir, was ihnen evtl. weniger wichtig erschien.
Reflexionen gehören nicht zu den üblichen Praktiken von Entwicklern. Es ist auch kein Standardvorgehen bei PP oder DPP. Diese Praktik entspringt der Forschungsmethode CAR (siehe Abschnitt 3.1) bzw. wurde vorher von einer anderen wissenschaftlichen Mitarbeiterin der AG SE bereits erfolgreich bei der Betreuung von PP-Sitzungen eingesetzt.
Da es sich um einen Termin ohne folgende Sitzung handelte, hatte es keine verspätende Wirkung auf andere Aktivitäten des DPP-Projektes. Eigentlich hatte es nur mich mehr Zeit gekostet, da ich auf die anderen warten musste. Ich konnte die Zeit jedoch für weitere Vorbereitungen auf die

Reflexion nutzen. Es kam dieses Mal und auch später oft vor, dass ich Fragebogen (Most Recent Session) von Entwicklern erst sehr kurz oder noch gar nicht für die Reflexion zur Verfügung hatte. Teilweise waren auch hier die verschiedenen Zeitzonen dafür verantwortlich. Ich las also den Fragebogen von Entwickler2, der mir erst an diesem Tag drei Stunden vorher von ihm zugeschickt wurde, noch einmal in Ruhe durch, vervollständigte dementsprechend meinen Leitfaden für die Reflexion und notierte mir andere Erkenntnisse, die ich daraus ziehen konnte, in meinen Aufzeichnungen.

Verspätungen zu Reflexionen II:

Zur Reflexion für Sitzung 10 kam Entwickler2 10 Minuten zu spät. Er gab keinen Grund an und wurde von den anderen beiden Entwicklern in meiner Anwesenheit auch nicht darauf angesprochen.
Da es sich um einen Termin ohne folgende Sitzung handelte, hatte es keine verspätende Wirkung auf andere Aktivitäten des DPP-Projektes. Auf solche Fälle wird weiter unten in Abschnitt 5.2.3 eingegangen. Es wurde allerdings auch in diesem Fall wieder Zeit verschwendet. Dieses Mal waren es sogar drei andere Personen, die warteten, denn außer Entwickler1 und mir war auch Entwickler3 für die Reflexion anwesend.
Die Verspätungen zu Reflexionen und die fehlende Interaktion der Entwickler untereinander verstärkten den Eindruck, dass sie ein falsches Bild über die Bedeutung der Reflexionen hatten. Sie sahen diese Termine nicht als Teil ihres Projektes, sondern ausschließlich als Termin zum Informationsaustausch mit mir. Entwickler2 stellte sogar die Notwendigkeit sich überhaupt außerhalb der Sitzungen zu unterhalten in Frage: „*we talk only when it's necessary, because talking is overhead ...*“
Der Etablierungsprozess wurde angepasst, indem Reflexionen auf Sitzungstage gelegt wurden. Ein Sitzungstag begann mit der Reflexion der vorherigen Sitzung. Damit sollten zusätzliche Termine, die als Mehraufwand angesehen wurden, vermieden werden. Die Hoffnung war, dass Reflexionen vor der Sitzung Verspätungen zu den Sitzungen verhindern könnten, stattdessen verschoben sich dadurch zusätzlich die Sitzungen.

Termin	Entwickler	Verspätung in Minuten	Grund	Probleme / Auswirkungen
Reflexion 3	1, 2	15	Termin vergessen	falsches Verständnis von Reflexionen
Reflexion 10	2	10	unbekannt	falsches Verständnis von Reflexionen

Tabelle 5.18 Zusammenfassung der Verspätungen der Entwickler zu Reflexionen

5.2.3.3 Verspätungen durch Reflexionen

Dreimal führten Reflexionen dazu, dass darauf folgende Sitzungen verspätet stattfanden (siehe Tabelle 5.19). Wie oben schon erwähnt fanden bis zur 7. Sitzung und für Sitzung 10 Reflexionen an anderen Tagen statt als die Sitzungen. Es stellte sich jedoch heraus, dass es leichter war, einen Termin für die Reflexion zu finden, wenn dieser am selben Tag wie eine Sitzung stattfand. Ein weiterer Vorteil, den diese Reflexionen hatten, war, dass sie so zeitnah an den Sitzungen waren, dass die Entwickler sich noch an die angesprochenen Sitzungs- oder Fragebogeninhalte erinnern konnten.

Termin	Entwickler	Verspätung in Minuten	Grund
Sitzung 8	alle Beteiligten	42	Reflexion vor der Sitzung
Sitzung 9	alle Beteiligten	19	Reflexion vor der Sitzung
Sitzung 12	alle Beteiligten	13	Reflexion vor der Sitzung

Tabelle 5.19 Verspätungen von Sitzungen durch Reflexionen

Im nächsten Abschnitt wird im Detail erläutert, warum die erste Anpassung des Etablierungsprozesses bzgl. Reflexionen noch nicht ausreichte, um Verspätungen im DPP-Projekt zu verhindern.

Verspätungen durch Reflexionen I

Wie die folgende Beobachtung zeigt gab es Situationen, in denen Reflexionen länger als erwartet dauerten. Dadurch verschoben sich auch die folgenden Aktivitäten.
Die 4. Reflexion dauerte 42 Minuten. Davon wurden die ersten 20 Minuten benötigt, um die Audioeinstellungen der technischen Infrastruktur anzupassen. Sitzung 8 konnte erst eine halbe Stunde später als geplant beginnen.
Da die Reflexionen vorher nur etwa 10 Minuten gedauert hatten, waren für diese Reflexion ebenfalls nur 15 Minuten eingeplant. Schlechte Vorbereitung der Entwickler und wichtige Anpassungen an der technischen Infrastruktur führten jedoch dazu, dass die folgende Sitzung erst sehr viel später als geplant beginnen konnte.

Die Sitzungen zuvor waren durch schlechte Audioqualität (metallische Nebengeräusche, Hintergrundgeräusche und Echos) belastet gewesen. In der Woche davor waren vom Saros-Team gerade die Einstellungen ermittelt worden, die dafür sorgen sollten, dass die Audioqualität in Zukunft besser würde. Obwohl die Teles-Entwickler bereits per Email Empfehlungen zu den Einstellungen bekommen hatten, war es nötig, die Audioeinstellungen von Mumble mit jedem einzeln und nacheinander noch einmal durchzugehen. Zum einen war dies erforderlich, weil die Entwickler ihre Einstellungen noch nicht geändert hatten. Zum anderen sind Einstellungen der Audioqualität immer abhängig von den geographischen und technischen Gegebenheiten. Die empfohlenen Einstellungen waren vorher in einer anderen Konstellation mit Teilnehmern des Saros-Teams getestet worden. Daher war es nötig, die Audioqualität zwischen allen vier Beteiligten der Sitzung noch ein wenig an die geänderte Umgebung anzupassen.

Da eine Verzögerung nicht eingeplant war, hatten die Entwickler danach nicht mehr genug Zeit, alle geplanten Aufgaben bis zum geplanten Sitzungsende zu erledigen. So hatte der verspätete Sitzungsbeginn die Auswirkung, dass Entwickler1 ausnahmsweise die Sitzung noch 20 Minuten in seine Mittagspause hinein verlängerte. Ein Indiz dafür, dass es im Projekt sehr wohl eine Rolle spielte, wie viel produktive Sitzungszeit am Ende übrig blieb.

Verspätungen durch Reflexionen II

Nach der 3. Reflexion war davon auszugehen, dass für die Sitzung nicht genug Zeit bleiben würde, um die geplanten Aufgaben zu erledigen, wenn die Reflexion innerhalb der geplanten Sitzungszeit stattfinden würde. Daher wurde ein Termin 15 Minuten früher vereinbart. Da es keine weiteren technischen Einstellungen mehr zu erledigen gab, hätten 15 Minuten ausreichen können.
Die Reflexion startete pünktlich und dauerte etwa 28 Minuten. Dieses Mal war der Grund für die Überlänge, dass die Entwickler zur Abwechslung einmal etwas ausführlicher bei ihren Antworten waren. Dies hatte sowohl positive als auch negative Aspekte. Ein positiver Gesichtspunkt war, dass die Entwickler sich mehr Zeit als üblich genommen hatten, um über ihre Erlebnisse in der

Sitzung zu sprechen. Demzufolge hatte Entwickler2 jedoch weniger Zeit für seine Mittagspause, denn der Zeitraum von 1,5 Stunden für die Sitzungen war genau zwischen die Mittagspausen der Entwickler gelegt worden - wieder ein Punkt, an dem die verschiedenen Zeitzonen eine Rolle spielten.
Die Reflexion dauerte wieder länger als die geplanten 15 Minuten, sodass der folgende Sitzungsbeginn sich deswegen verspätete. Jedoch hatte es dieses Mal keine Auswirkungen auf das Ende der Sitzung, denn die geplanten Aufgaben konnten schneller als erwartet erledigt werden.

Verspätungen durch Reflexionen III

Als die Reflexion zur 11. Sitzung wieder für eine Verspätung des Sitzungsbeginns sorgte, war es nötig, den Etablierungsprozess erneut anzupassen, damit Verspätungen durch Reflexionen in Zukunft weniger Einfluss auf die produktive Sitzungszeit haben würden.
Wie vorher bereits erwähnt war eines der zentralen Punkte meiner Tätigkeit als Betreuerin, dafür zu sorgen, dass die Sitzungszeit produktiv genutzt werden konnte und die Entwickler sich durch den Etablierungsprozess (also auch die Reflexionen) so wenig wie möglich gestört fühlten. Die Entwickler hatten bereits geäußert, dass sie die Sitzungen in Verbindung mit dem Etablierungsprozess (Fragebogen und Reflexionen) als zu großen, zusätzlichen Mehraufwand betrachteten.
Aufgrund dessen wurde versucht eine weitere Zeitersparnis im Etablierungsprozess vorzunehmen. Ab Sitzung 12 wurden Reflexionen für Sitzungen einer ganzen Woche nur noch am Ende der letzten Sitzung durchgeführt (und später auch die Fragebogen gekürzt).
Ausgerechnet die letzte Sitzung dieser Woche fiel spontan aus, sodass keine Reflexion für Sitzungen in dieser Woche stattfand. Es war nicht möglich, die Reflexion zu einem späteren Zeitpunkt nachzuholen, da die Entwickler sich nach mehr als einem Tag Abstand meistens nicht mehr an die Inhalte erinnern konnten.

Termin	Entwickler	Verspätung in Minuten	Grund	Probleme/ Auswirkungen
Sitzung 8	alle Beteiligten	42	Reflexion vor der Sitzung	fehlende Vorbereitung
Sitzung 9	alle Beteiligten	19	Reflexion vor der Sitzung	keine Auswirkungen
Sitzung 12	alle Beteiligten	13	Reflexion vor der Sitzung	Anpassung der Reflexion

Tabelle 5.20 Zusammenfassung der Verspätungen von Sitzungen durch vorherige Reflexionen

5.2.3.4 Zusammenfassung

Verschiedene Ausprägungen von Verspätungen und ihre Bedeutung für den DPP-Prozess wurden anhand von Kontext, Gründen und Konsequenzen im vorangehenden Abschnitt aufgezeigt. Man kann diese Beobachtungen nun zusammenfassen, indem ihre Relevanz bezogen auf DPP erörtert wird.
Tabelle 5.21 fasst die Verspätungen der Entwickler entsprechend der obigen Erörterung zusammen.

Termin	Entwickler	Verspätung in Minuten	Grund	Probleme/ Auswirkungen
Sitzung 5	2	17	unbekannt	konfliktscheu, falsches Rollenverständnis
Sitzung 7	2	23	unbekannt	Rolle des dritten Entwicklers bei DPP
Sitzung 11	2	15	unbekannt	konfliktscheu, falsches Rollenverständnis
Sitzung 13	2	17	Mittagspause	konfliktscheu, falsches Rollenverständnis
Reflexion 3	1, 2	15	Termin vergessen	falsches Verständnis von Reflexionen
Reflexion 10	2	10	unbekannt	falsches Verständnis von Reflexionen
Sitzung 8	alle Beteiligten	42	Reflexion vor der Sitzung	fehlende Vorbereitung
Sitzung 9	alle Beteiligten	19	Reflexion vor der Sitzung	keine Auswirkungen
Sitzung 12	alle Beteiligten	13	Reflexion vor der Sitzung	Anpassung der Reflexion

Tabelle 5.21 Übersicht über Verspätungen von Entwicklern zu Sitzungen, zu Reflexionen und durch Reflexion im Etablierungsprozess

Verspätungen verkürzten die Sitzungszeit teilweise um bis zu 20 %. Zwar hatten die Verspätungen meistens keine Auswirkungen auf die Erreichung der Sitzungsziele, doch in der Summe ging dadurch viel Zeit für das gesamte Projekt verloren (siehe auch 5.3.1).
Aufgrund der Voraussetzungen für eine vorteilhafte Anwendung von DPP konnten Sitzungen erst beginnen, wenn mindestens zwei Entwickler (Entwickler1 und einer der Entwickler aus Bangalore) anwesend waren. Daher verkürzte sich die produktive Sitzungszeit bereits dadurch, dass einer der beteiligten Entwickler sich zur Sitzung verspätete. Besonders weil das Sitzungsende auf Wunsch von Entwickler1 (der dann mit seinen Kollegen Mittagspause machen wollte) fest war und sich außer in Ausnahmefällen nicht beliebig verschieben ließ.
Die Analyse zeigte, dass einer der Entwickler erheblich zu spät zu mehreren Sitzungen erschien. Eine Vermutung bzgl. der Ursache ist, dass der Entwickler aufgrund von fehlenden Zielvorstellungen für bestimmte Sitzungsinhalte und einem falschen Rollenverständnis andere Prioritäten setzte, und dadurch auch kein Verantwortungsbewusstsein für die aktive Teilnahme bzw. Pünktlichkeit in den Sitzungen vorhanden war.
Weiterhin ist es möglich, dass die Reaktion der anderen Entwickler, die sich nicht über die Verspätungen äußerten, (besonders bei Sitzungen zu dritt) dazu führte, dass sich das falsche Rollenverständnis des Entwicklers noch verstärkte.
Eine weitere Vermutung ist, dass der betreffende Entwickler seine Position im Team durch die Verspätungen testen wollte. Dies würde auch erklären, warum er nach der Ermahnung des projektverantwortlichen Entwicklers nicht mehr zu spät kam.
Die Reaktion der Entwickler zeigt, dass sie vermieden über konfliktbehaftete Themen zu sprechen. Insgesamt fehlte es an Offenheit in der Kommunikation der Beteiligten. Daher konnten auch die Gründe für die Verspätungen nicht abschließend geklärt werden.
Es stellte sich heraus, dass zusätzliche Maßnahmen (Reflexionen), die den DPP-Prozess unterstützen sollten, für die Entwickler ungewohnt waren. Sie sahen den Etablierungsprozess eher als Mehraufwand denn als Mehrwert an. Dadurch räumten sie den Aktivitäten weniger hohe Priorität ein, was sich durch Verspätungen zu Reflexionen und verzögerte Abgabe der Fragebogen äußerte. Dies hatte nicht nur negative Auswirkungen. Bei Verspätungen zu Reflexionen war es möglich, die Zeit für weitere Vorbereitungen, z.B. zur Auswertung verspätet eingegangener Fragebogen, zu nutzen.
Durch mehrfache Anpassung des Etablierungsprozesses (Terminvereinbarung an Sitzungstagen nach Sitzungen) konnten durch Reflexionen verursachte Verspätungen vermieden werden. In

diesem Zusammenhang hatte sich gezeigt, dass die verschiedenen Zeitzonen sowohl für die Terminplanung von Reflexionen, als auch für die Abgabe der Fragebogen eine Rolle spielte.
Verspätungen durch Reflexionen hatten nicht nur negative Effekte auf die Sitzungszeit, wenn bspw. die geplanten Aufgaben schneller als erwartet erledigt werden konnten oder wenn der Grund für die verlängerte Reflexion eine Maßnahme beinhaltete, die für das gesamte DPP-Projekt von Vorteil war (gemeinsame Audioeinstellungen). Zeitnahe Reflexionen, die die Sitzungszeit verkürzten, hatten den Vorteil, dass die Terminplanung durch anschließende Sitzungen leichter war und die Entwickler sich noch an die Geschehnisse der Sitzung und die Angaben im Fragebogen erinnern konnten.

5.2.4 Vorkommen und Fehlen von Awareness

Der Begriff Awareness wurde bereits in Abschnitt 2.2 eingeführt. Es handelt sich dabei um das Bewusstsein von Personen über ihre eigenen Aktivitäten im Kontext zu Aktivitäten ihrer Umgebung in verteilter kollaborativer Softwareentwicklung. Awareness ist wie in Abschnitt 2.2 bereits beschrieben wichtig für DPP. Sie unterstützt die koordinierte Zusammenarbeit und gewährleistet damit, dass manche Vorteile von PP, z.B. Pair-Review, auch bei DPP vorhanden sein können.
Es wurden vier für DPP-Sitzungen relevante Formen von Awareness vorgestellt. Dies sind soziale, informelle, gruppenstrukturelle und Workspace Awareness (siehe Abbildung 2.2). Außerdem wurden verschiedene Fragen vorgestellt, die Elemente von Awareness beschreiben (siehe Tabelle 2.1).
Es hat sich in verschiedenen Studien gezeigt, dass durch Awareness die Zusammenarbeit einer Gruppe signifikant effektiver sein kann als individuelle Arbeit [GGR96], und dass das Erhöhen der Awareness sich positiv auf DPP auswirkt [GGR96].
Im Verlauf des Projektes konnten sowohl positive Effekte von Awareness als auch negative Effekte von fehlender Awareness beobachtetet werden. Im folgenden Abschnitt soll die Bedeutung von Awareness für einen DPP-Prozess anhand von konkreten Phänomenen erläutert werden.

5.2.4.1 Positive Auswirkungen von Awareness

In den Sitzungen des Etablierungsprozesses waren immer wieder Vorteile von Awareness bei DPP zu beobachten. Beispielsweise war meistens ausreichend Awareness vorhanden, wenn Entwickler1 Driver war. Ein Auszug aus der 12. Sitzung (in vier Ausschnitten) zeigt, wie sich ein hohes Maß an Awareness positiv auf die Zusammenarbeit auswirken kann.

Auschnitt 1:
Entwickler1 ist Host und Driver. Er fragt Entwickler2, ob sich eine Datei auch bei ihm geöffnet hat: *"has it opened on your side"*. Einige Minuten später teilt Entwickler1 mit, dass er sich in eine andere Datei begibt: *„I switch back to [Dateiname]*".
Entwickler2 bestätigt mit „*mhm"* und *„yep"*, dass er gehört hat, was Entwickler1 sagte.

Ausschnitt 2:
Entwickler1 kommentiert seine weiteren Handlungen: *"I will take the whole function and remove everything we don't need"* und später *"I think we have removed most of the stuff we do not need."*

Ausschnitt 3:
Entwickler2 hält die Erklärungen von Entwickler1 an und benutzt eine Zeilennummer, um nach einer bestimmten Stelle in einer Datei zu fragen: *„one second, here we have in line no. 500, what is an event?*"

Ausschnitt 4:
Entwickler2 macht Entwickler1 darauf aufmerksam, dass er dabei ist, in der falschen Datei zu kodieren: „*This is in [Dateiname]*“, und Entwickler1 bedankt sich für den Hinweis: *„Thanks for your hint.*“

Diese Ausschnitte zeigen auf verschiedene Arten, wie Awareness bei DPP behilflich sein kann. Im ersten Ausschnitt half Entwickler1 Entwickler2 dabei, sich im Workspace zu orientieren. Hierbei wurde er durch den Verfolgermodus und andere Awareness-Funktionalitäten von Saros unterstützt. Im zweiten Ausschnitt kommentierte Entwickler1 seine Handlungen und gab gleichzeitig Erklärungen für Entwickler2 dazu ab. Im dritten Ausschnitt zeigte Entwickler2 durch eine gezielte Frage sein Interesse für die Aktivität von Entwickler1 und verwendete die Zeilennummer in der gerade bearbeiteten Datei, um auf die Stelle zu deuten, auf die er sich bezog. Im vierten Ausschnitt hinterfragte Entwickler2 die Stelle im Workspace, an der die Änderungen vorgenommen werden sollten, und machte damit Entwickler1 auf dessen falsche Position aufmerksam. Der daraus entstandene Vorteil für DPP war, dass Entwickler2 auf diese Weise die Erzeugung eines Defekts sowie zusätzlichen Zeitaufwand für spätere Korrekturen vermieden hatte.
Alle Ausschnitte können unter verschiedenen Arten von Awareness betrachtet werden. Eine Möglichkeit ist, sie als Form von Workspace Awareness einzuordnen, unter dem Aspekt der Orientierung im Workspace. Ausschnitte 1 und 3 enthalten jedoch auch Elemente von Interessensbekundungen, sodass sie auch als soziale Awareness bezeichnet werden können.
Wie aus den Ausschnitten deutlich wird, gibt es verschiedene Formen von Awareness, die gleichzeitig auftreten können. Awareness wirkt sich, wie aufgezeigt wurde, positiv auf die Zusammenarbeit aus. Daher ist es wünschenswert, so viel wie möglich davon bei DPP einzusetzen. Welche Auswirkungen fehlende Awareness haben kann, zeigt der nächste Abschnitt.

5.2.4.2 Nachteile durch fehlende Awareness

Die weiteren Analysen betrachten den umgekehrten Fall: Welche Auswirkungen können auftreten, wenn manche oder gleich mehrere Elemente der Awareness bei DPP entfallen. Es werden schrittweise verschiedene Phänomene aufsteigend nach der Schwere ihrer Auswirkungen auf die Sitzung erörtert.
Die ersten beiden Beobachtungen zu Workspace Awareness (Location und Identity) zeigen, dass es nicht immer nötig ist, alle Elemente von Awareness zu unterstützen, damit eine reibungslose Zusammenarbeit funktionieren kann. Es wird aber auch dargestellt, dass evtl. später noch Auswirkungen auftreten können.
In mehreren Sitzungen musste der Observer den Driver fragen, an welcher Stelle sich dieser im Workspace gerade befindet oder womit dieser sich gerade beschäftigt. Obwohl Saros-Funktionalitäten vorhanden gewesen wären, die ihn bei der Verfolgung des Drivers (Verfolgermodus, Highlighting und verschiedene Annotations, siehe Abschnitt 2.5) automatisch unterstützt hätten.
Anders als bei den ersten Beobachtungen zeigten sich schon deutlich negativere Auswirkungen auf den Sitzungsablauf, wenn verschiedene Awarenessformen (Actions) gleichzeitig entfielen. Man konnte die Ursache schnell durch fehlende Workspace und soziale Awareness erklären. Dennoch wurde dadurch der Arbeitsfluss vollständig für mehrere Minuten unterbrochen.
Es kam bspw. in einer der Sitzungen nach einem Rollenwechsel von Driver und Observer vor, dass erst mehrere Minuten vergingen, bevor die Beteiligten Inkonsistenzen feststellten, da der Driver zu wenig kommentiert hatte.

Wohl am schwerwiegendsten erwiesen sich jedoch die Probleme, die sich durch das Fehlen von gruppenstruktureller Awareness (Expectations und Abilities) beschreiben lassen. In diesen Sitzungen zeigten sich Probleme in Verbindung mit den Rollen und den Verantwortlichkeiten der

beteiligten Entwickler. Die Entwickler zeigten in mehreren Sitzungen, dass sich ihre Erwartungen bezüglich der Verteilung der Driver-Rolle unterschieden.
Im Folgenden wird im Detail erläutert, wie sich Elemente von Awareness in den Sitzungen zeigten und welche Auswirkungen dies auf die DPP-Sitzungen hatte. Zur Beschreibung der einzelnen Phänomene werden zusätzlich Elemente der Awareness aus Tabelle 2.1 verwendet.

Identity

Identity (dt. Identität) ist ein Element von Awareness, das sich mit der Frage nach der Präsenz von anderen Personen im Workspace beschäftigt. Dieses Element spielt eine entscheidende Rolle für das Zustandekommen einer Sitzung, weil es sich auf die Anwesenheit und die Erreichbarkeit einer Person im gemeinsamen Arbeitsumfeld bezieht.
Es handelt sich dabei um eine Form der informellen Awareness. In Teams, die nicht verteilt arbeiten, wird informelle Awareness durch einen gemeinsamen Arbeitsplatz unterstützt. In den DPP-Sitzungen wird dies durch die technische Infrastruktur mit Mumble und Saros geleistet. Die Audiokommunikation über Mumble zeigt bspw. in einem Fenster an, wer sich noch auf dem Kommunikationskanal angemeldet hat. Saros signalisiert die Anwesenheit anderer Benutzer, die zur selben Zeit mit dem Server verbunden sind, über eine farbliche Hervorhebung in der Liste der *Buddies* (anderer bekannter Benutzer) im Roster (siehe Abschnitt 2.5 bzw. Screenshot in Anhang D).
Wie in der Analyse zu ausgefallenen Sitzungen bereits erläutert (siehe Abschnitt 5.2.2), ist es bei DPP meistens notwendig, sich gegenseitig darüber zu informieren, ob und wann sich die Beteiligten zur Arbeit zusammenfinden bzw. wann sie nicht an Sitzungen teilnehmen können. Es muss jedoch nicht immer problematisch für den Verlauf einer Sitzung sein, wenn dieses Element von Awareness fehlt. Die Analyse zu ausgefallenen Sitzungen zeigte, dass es wichtig ist, über die Präsenz von anderen Beteiligten sicher sein zu können, wenn es sich dabei um essentielle Beteiligte handelt. Sie zeigte aber auch, dass ein dritter Beteiligter den Sitzungen fernbleiben konnte, ohne den Sitzungsablauf dadurch zu behindern.
In der Analyse zu Verspätungen wurde erläutert (siehe Abschnitt 5.2.3), dass eine DPP-Sitzung erst beginnen kann, wenn die dafür notwendigen Beteiligten alle anwesend sind. Daher ist es entscheidend, dass die Beteiligten ihre Anwesenheit signalisieren, sobald sie zur Verfügung stehen. Wie bei der Analyse zu ausgefallenen Entwicklern zeigte sich auch bei der Analyse zu Verspätungen, dass fehlende Awareness über das Fernbleiben eines dritten Entwicklers den Sitzungsverlauf nicht unbedingt beeinflusst.

Location

Location (dt. Standort) ist ein Element von Awareness, das sich mit der Frage beschäftigt, an welcher Stelle im Workspace eine Person gerade aktiv ist (siehe Abbildung 2.1). Es gehört damit zur Workspace Awareness.
Obwohl Saros Workspace Awareness durch eine Vielfalt von technischen Funktionalitäten, z.B. dem Verfolgermodus und farblichen Zeilenmarkierungen, unterstützt (siehe Abschnitt 2.5), ist dadurch noch keine Workspace Awareness garantiert. Auf die Distanz und ohne Einsicht in die Aktivitäten des anderen, z.B. durch Desktop Sharing, kann nicht unterschieden werden, ob das Angebot der Awareness-Funktionalitäten wirklich genutzt wird oder nicht. Der Driver kann also nicht sicherstellen, dass der Observer ihm auch wirklich folgen kann.

Tatsächlich zeigte sich an manchen Stellen, dass die Entwickler die zusätzlichen Funktionalitäten gar nicht einsetzten. Beispielsweise kam es öfter vor, dass der Observer den Driver fragen musste, in welcher Datei dieser sich gerade befindet, weil er nicht den Verfolgermodus verwendete, der ihn automatisch an die Position des Cursors des Drivers gesetzt hätte. Wie sich in den vorher beschriebenen Ausschnitten der 12. Sitzung außerdem beispielhaft zeigte, verwendeten die Entwick-

ler lieber den Dateinamen und die Zeilennummer, um sich auf konkrete Positionen im Code zu beziehen, als dies mit farblichen Markierungen durch Bewegen ihres Cursors zu signalisieren.
Es kann für eine DPP-Sitzung gravierend sein, wenn die Entwickler gegenseitig nicht wissen, an welcher Stelle sich der andere gerade befindet, denn viele Vorteile, z.B. Pair-Review, die PP normalerweise aufweisen kann, werden dadurch nicht genutzt.
Da die Entwickler es durch ihre Variante mit Zeilennummer und Dateinamen trotzdem geschafft haben, regelmäßig ihre Position im Code gegenseitig zu finden, hatten diese Situationen scheinbar keine negativen Auswirkungen auf die Effektivität der Sitzungen.
Trotzdem zeigten sich in späteren Sitzungen als Konsequenz Unsicherheiten über den Fokus (siehe Abbildung 2.1) des Observers. Es zeigte sich, dass Entwickler1 sich nicht mehr sicher war, ob ihm der Observer wirklich folgte. In den letzten Sitzungen hatte Entwickler1 das Gefühl, anhand der Verfolgung der Cursor-Bewegungen des Observers sicherstellen zu müssen, dass dieser ihm auch wirklich folgte, obwohl die Position des Cursors alleine darüber gar keine Informationen liefern kann.

Actions

Actions (dt. Aktivitäten) sind ein Element von Awareness, das sich mit der Frage beschäftigt, was Personen gerade tun. Damit sind sie ein weiteres Element der Workspace Awareness.
Es gibt harmlose Fälle, in denen sich fehlende Awareness über Aktivitäten im Workspace nicht bemerkbar macht. Beispielsweise war dies der Fall, wenn der Driver statt mit seinem Cursor Stellen zu markieren nur den Mauszeiger über den Bildschirm bewegte. Die fehlende Awareness (in Form der visuellen Unterstützung durch den Cursor) konnte er jedoch durch seine Erklärungen kompensieren, sodass dem Observer keine wichtigen Informationen vorenthalten wurden.
Die Kombination von mehreren entfallenen Awareness Elementen konnte jedoch in Sitzungen sehr wohl zu Problemen führen. In der 5. Sitzung wartete der Observer bspw. darauf, dass der Driver drei kleinere Methoden implementieren würde. Erst nachdem der Driver einige Minuten später fragte: *Is it ok?*“, merkte der Observer, dass die Änderungen im Code des Drivers bei ihm nicht angezeigt wurden. In diesem Fall kam zu der fehlenden Workspace Awareness noch fehlende soziale Awareness hinzu. Weder kommentierte der Driver seine Aktionen während der Implementierung noch signalisierte der Observer, z.B. durch Nachfragen, dass er auf weitere Aktionen wartete.
Die aufgetretenen Probleme wurden durch technisches Versagen in Kombination mit fehlender Kommentierung herbeigeführt. Es war den Entwicklern aus technischer Sicht nicht möglich, die Inkonsistenz früher zu bemerken, denn aufgrund des Entwicklungsstandes von Saros wurden zu diesem Zeitpunkt keine Inkonsistenzen angezeigt. Allerdings hätten die darauf folgende Unterbrechung der Sitzung von mehreren Minuten und der Informationsverlust vermieden werden können, wenn der Driver seine Handlungen gleich kommentiert hätte. Die Verbalisierung der eigenen Aktivitäten wird von Salinger und Prechelt [SP08] als *Think Aloud* (dt. laut Denken) bezeichnet. *Think Aloud* ermöglicht dem Observer bei PP und bei DPP, den Aktivitäten des Drivers leichter zu folgen.
Durch den Ausfall von zwei Awareness Elementen kam es also zu einem kurzzeitigen Informationsausfall und Zeitverlust in der Sitzung. Da der Driver meistens gut kommentierte, waren solche Formen der fehlenden Awareness selten.

Expectations und Abilities

Expectations und *Abilities* (dt. Erwartungen und Fähigkeiten), sind Elemente von Awareness, welche die Frage danach stellen, was als nächstes von Personen bei DPP erwartet wird und welche dieser Erwartungen sie erfüllen können. Diese Elemente gehören in den Bereich der gruppenstrukturellen Awareness, wenn dafür ein bestimmtes Rollenverständnis Voraussetzung ist.
Wie in den vorherigen Analysen zu Zielen, Ausfällen und Verspätungen wird nun die Rollenverteilung zwischen den Entwicklern noch einmal von einer anderen Perspektive erörtert, die sich

stärker auf Erwartungen und Verantwortlichkeiten bezieht. Bisher noch ungeklärt geblieben ist die Frage, warum mehrere Versuche von Entwickler1, Entwickler2 die Driver-Rolle zu übergeben, scheiterten. Leider gibt es bisher nur wenige Studien, die sich damit beschäftigen, was innerhalb von PP- bzw. DPP-Sitzungen geschieht oder geschehen sollte. Es gibt bspw. auch keine Empfehlungen dazu, wie und wann Rollenwechsel bei PP oder DPP stattfinden sollten. Eine Studie sagt, dass regelmäßige Rollenwechsel zwischen Driver und Observer zwar nicht vorgeschrieben, aber durchaus üblich sind [Han04]. Andere Studien belegen, dass sehr gute Ergebnisse auch ohne Rollenwechsel erzielt wurden [SL08].

Sitzung 10 war bereits die fünfte Sitzung vom Typ Pair-Developing im DPP-Projekt, und Entwickler1 hatte als Kommentar zur vorherigen Pair-Developing-Sitzung bereits im Fragebogen angegeben: *„I think I spent too much time in driver role this session"* Er versuchte daher, Entwickler2 die Driver-Rolle zu übergeben. Entwickler2 reagierte auf die Aufforderung mit verlegenem Lachen, sodass Entwickler1 Driver blieb. In der Reflexion zur Sitzung beschrieb Entwickler1 die Situation folgendermaßen: *„[Entwickler2] was not familiar with the class [Klassenname]. I created the object to give him the hint he needed to do the ongoing work."*

Bei der beschriebenen Situation konnte bis zuletzt nicht geklärt werden, welche Gründe dazu führten und welche Auswirkungen sie hatte. Die Zuordnung dieses Phänomens in fehlende Awareness ist nur eine der Möglichkeiten, dieses zu interpretieren. Zur Unterstützung der Annahme, dass es sich hierbei tatsächlich um einen Fall von fehlender Awareness handelte, sind zwei Punkte zu klären. Zum einen ist zu klären, ob Entwickler2 wusste, was Entwickler1 von ihm erwartete (Expectations). Zum anderen ist zu klären, ob er die Möglichkeit hatte, die Erwartungen auch zu erfüllen (Abilities).

Der erste Punkt, der die Theorie über fehlende Awareness unterstützen würde, müsste belegen, dass die Entwickler sich über Rollen und Erwartungen nicht bewusst waren bzw. dass dies für Entwickler2 gilt. Es wurde bereits als Ergebnis der Analyse in Abschnitt 5.2.1 dargelegt, dass die Entwickler sich eben nicht über prozessuale Ziele abgestimmt hatten, und daher die Rollen und Verantwortlichkeiten innerhalb der Sitzungen nicht bekannt waren. Die Frage ist, ob Entwickler2 vielleicht trotzdem wusste, was für eine Rolle Entwickler1 ihm zugedacht hatte und welche Erwartungen damit einhergingen.

Oberflächlich betrachtet ist dies kein Fall von fehlender Awareness, denn Entwickler2 hatte schon verstanden, dass er in diesem Augenblick die Driver-Rolle übernehmen soll, diese aber nicht angenommen. Jedoch ist das Problem damit noch nicht gelöst, denn wie bereits in Abschnitt 5.2.3 erläutert hatte Entwickler2 kein Vorwissen über PP oder DPP und konnte daher auch nicht wissen, dass es üblich ist, bei PP und DPP regelmäßig die Rollen zu wechseln. Zusammen mit der Erkenntnis, dass Entwickler1 seine Vorstellungen darüber nicht mit Entwickler2 teilte, ist zu vermuten, dass Entwickler2 nicht bewusst war, warum er als Novize in dieser Situation die Driver-Rolle übernehmen sollte.

Der zweite Punkt, der die Theorie über fehlende Awareness unterstützen würde, müsste zusätzlich noch belegen, dass Entwickler2 auch nicht die Möglichkeit hatte, die in ihn gesetzten Erwartungen zu erfüllen. Hier gibt es zwei Möglichkeiten, von der nur eine dafür sprechen würde, dass dieses Phänomen wirklich eine Form von fehlender Awareness war. Im anderen Fall wäre es kein Missverständnis über das Rollenverständnis, sondern einfach persönliche Bequemlichkeit gewesen, die zu Konflikten bzgl. der Driver-Rolle führte. Entwickler2 machte mir gegenüber mehrere Aussagen, die beide Annahmen unterstützen würden.

Zwei Gründe sprechen dafür, dass Entwickler2 sich nicht im Stande sah, die Driver-Rolle zu übernehmen: (1) weil er davon ausging, dass es nicht in seine Verantwortung fiel und (2) weil ihm das nötige Wissen fehlte.

Es war verwunderlich, dass Entwickler2 darauf verzichtete, die Driver-Rolle zu übernehmen. Die meisten Studien über PP und DPP gehen davon aus, dass jeder Beteiligte gerne Driver sein möchte. Dies gilt auch für Entwickler, die als Novize an PP- bzw. DPP-Sitzungen teilnehmen [SR03].

Entwickler2 bestätigte, dass auch er gerne Driver gewesen wäre, dass er aber davon ausging, dass dieses Privileg dem ursprünglichen Autor des Codes überlassen ist. Er sagte: „*Because he [Entwickler1] is the owner, he has the driver role.[...] If I am the owner I will be the driver.*"

Daraus lässt sich in mehrfacher Hinsicht schließen, dass Entwickler2 ein falsches Verständnis für seine Möglichkeiten bzw. Verantwortungen in einer DPP-Sitzung hatte. Zum einen sind alle Ent-

wickler in einer Pair-Developing-Sitzung bereits durch ihre Anwesenheit im gleichen Maße verantwortlich für die darin erzeugten Artefakte. Unabhängig davon wie viel sie durch eigene Eingabe an der Tastatur dazu beigetragen haben, gelten sie dadurch als Autor des entstandenen Quellcodes. Aus demselben Grund bestimmt sich auch die Aufteilung der Driver- und Observer-Rollen in einer Pair-Developing-Sitzung nicht danach, wer bereits durch Eingabe an der Tastatur mehr zum Ergebnis beigetragen hat.
Weiterhin gab Entwickler2 in der Reflexion an, dass ihm das nötige Wissen über den vorhandenen Code noch fehlte. Dies wurde indirekt auch durch die vorbereitende Maßnahme von Entwickler1 bestätigt, der nicht davon ausging, dass es Entwickler2 ohne diese möglich wäre, die Implementierung als Driver fortzuführen.

Es lässt sich an dieser Stelle der Schluss ziehen, dass es sich um fehlende Awareness handelte, weil Entwickler2 die Erwartungen von Entwickler1 nicht unmittelbar erfüllen konnte.
Es ist jedoch auch möglich, dass all die vorher erörterten Annahmen falsch sind und Entwickler2 wusste, wie er als Driver weiter vorgegangen wäre. In einem Gespräch darüber in der folgenden Reflexion machte er zwei Aussagen, die diese Erklärung unterstützen könnten. Zuerst sagte er: *I was not fully confident what to write.“* Zudem beschrieb er seine Reaktion wie folgt: *„A remote person will think I am lazy...*“. In diesem Fall wäre es kein Problem von fehlender Awareness gewesen, wenn er z.B. nur aus persönlicher Bequemlichkeit oder aus Angst sich vor einem Entwickler mit mehr Domänenwissen zu blamieren die Driver-Rolle abgelehnt hatte.

Die vorhandenen Daten lassen sowohl den Schluss zu, dass es sich um fehlende Awareness handelt als auch, dass es ganz andere Gründe gehabt haben könnte. Da die Entwickler nur sehr wenige Informationen darüber preisgaben und noch einige Punkte ungeklärt blieben, kann weder endgültig beurteilt werden, ob es sich dabei tatsächlich um einen Fall von fehlender Awareness handelte, noch kann erläutert werden, welche Auswirkungen sich daraus genau ergaben.
Die Entwickler beschlossen, trotz der Konflikte keine Regeln für Rollenwechsel festzulegen, sondern wollten lieber spontan darüber entscheiden, was dazu führte, dass Entwickler2 bis auf wenige Ausnahmen Observer in den Sitzungen blieb.
Es könnte vermutet werden, dass dadurch die Sitzungen für beide Entwickler anstrengender als nötig waren. Entwickler1, der ständig die Driver-Rolle hatte, musste dauerhaft Eingaben an der Tastatur vornehmen und auch ununterbrochen reden bzw. erklären. Entwickler2 musste in seiner Rolle als Observer ständig beobachten und durfte nie selbst Code erzeugen, was evtl. seine Aufmerksam senkte, weil er sich öfter langweilte.

5.2.4.3 Zusammenfassung

Wie oben im Detail erläutert, kann Awareness in verschiedenen Formen in DPP-Sitzungen erscheinen und unterschiedlich starke Auswirkungen auf den Ablauf haben. Tabelle 5.22 fasst alle vorher beschriebenen Phänomene noch einmal zusammen. Der Kontext beschreibt weitere Faktoren, die auf die Phänomene Einfluss hatten.

Awareness -Typ	Gründe	Kontext	Auswirkungen
Identity (informelle Awareness)	Fernbleiben ohne Absage	Beteiligung als dritter Entwickler	keine direkten Auswirkungen auf den Sitzungsverlauf
Location (Workspace Awareness)	räumliche Distanz, Unabhängigkeit des Observers durch die techn. Infrastruktur	keine Verwendung der techn. Awareness Unterstützung	weniger Awareness als bei PP
Actions (Workspace und soziale Awareness)	technische Probleme, fehlende Kommentierung	mehrere Formen von fehlender Awareness treten gleichzeitig auf	technische Probleme bleiben länger unerkannt
Expectations, Abilities (gruppenstrukturelle Awareness)	persönliche Bequemlichkeit oder unterschiedliches bzw. falsches Rollenverständnis	wenige Rollenwechsel in Sitzungen	unvorteilhafte Rollenverteilung

Tabelle 5.22 Übersicht über Formen von fehlender Awareness im Etablierungsprozess

Es können verschiedene technische Hilfsmittel eingesetzt werden, um die Awareness von PP-Sitzungen nachzuahmen. Jedoch ist nicht sicher, dass diese auch wirklich eingesetzt werden. Daher können Vorteile von räumlicher Nähe in PP-Sitzungen für DPP-Sitzungen nicht bedingungslos gewährleistet werden. Auch bei PP kann der Observer abschalten und Fehler übersehen, doch hat der Driver durch die Anwesenheit im selben Raum leichter die Möglichkeit, dies auch mitzubekommen.
Die Driver-Rolle erfordert, dass der Driver für den Observer seine Aktivitäten kommentiert. Sonst kann der Observer den Aktivitäten evtl. nicht folgen, und technische Probleme werden erst später als nötig entdeckt.
Wenn sich die Entwickler nicht über ihre Einstellung und ihre Erwartungen zu DPP-Sitzungen austauschen, kann es dazu führen, dass die Beteiligten mit der Verteilung der Rollen und der Verantwortlichkeiten unzufrieden sind bzw. die Rollen nicht immer vorteilhaft aufgeteilt werden.

Zusammenfassend ist festzuhalten, dass es in Fällen von Workspace oder informeller Awareness teilweise möglich ist, durch Anpassungen der technischen Infrastruktur, z.B. durch zusätzliche Hinweise an den Driver über aktive Fenster des Observers, weitere Unterstützung der Awareness zu bieten und fehlende Awareness zu verhindern. Probleme von sozialer oder gruppenstruktureller Awareness zu verhindern ist schwieriger. Diese können teilweise durch zusätzliche Regeln wie z.B. *Think Aloud* des Drivers vermindert werden, doch kann nicht endgültig garantiert werden, dass sie nicht mehr vorkommen. Grundsätzlich ist es besonders schwierig, Probleme in der Zusammenarbeit mit DPP zu vermeiden, falls mehrere Formen der Awareness gleichzeitig entfallen.

5.3 Erkenntnisse des Etablierungsprozesses

Die Basis für Auswertungen über den Etablierungsprozess lieferten Daten aus den Befragungen (Fragebogen, Einzelgespräche und Reflexionen) sowie Beobachtungen aus den Sitzungen. Anhand dessen konnte ich Geschehnisse des Entwicklungs- und Etablierungsprozesses nachvollziehen und Einschätzungen der Entwickler bezüglich des Erfolges der Sitzung während des Projektes messen. Die Reflexionen hatten eine besondere Rolle im Etablierungsprozess. Sie dienten vor allem dem Informationsaustausch außerhalb der Sitzungen und der Einführung weiterer optimierender Maßnahmen.

5.3.1 Übersicht über aufgetretene Probleme

Die folgende Liste fasst gravierende Probleme, die während des Etablierungsprozesses aufgetreten sind, zusammen.

1. Probleme mit der technischen Infrastruktur (Eclipse, OpenVPN, Mumble) traten auf. Besonders Schwächen von Saros waren am Anfang problematisch. Die Software war anfangs noch eine α-Version, und die Defekte wirkten sich negativ auf die Geschwindigkeit und die Zuverlässigkeit aus.
2. Einige Sitzungsausfälle führten zu längeren Pausen zwischen den Sitzungen.
3. Am Anfang von Sitzungen gab es Verzögerungen durch Verspätungen und lange Vorbereitungsphasen (siehe Abbildung 5.4), weil die Routine im Umgang mit der technischen Infrastruktur (besonders Saros) oder die Vorbereitung auf die Sitzung fehlte.
4. Die Entwickler hatten Vorurteile gegenüber geeigneten Sitzungstypen.
5. In manchen Sitzungen und Reflexionen kam es vor, dass die Entwickler durch äußere Einflüsse abgelenkt waren.
6. In der Zusammenarbeit der Entwickler fehlte teilweise direkte Kommunikation bei der Abstimmung zu Sitzungen, während der Sitzungen und in Reflexionen
7. Es zeigte sich zudem ein Konflikt der Rollenverteilung in Pair-Developing-Sitzungen.
8. Teilweise fehlte den Entwicklern in den Sitzungen Awareness. Dem Driver fehlten Informationen über die Aktivitäten des Observers und umgekehrt.

Besonders das erste Drittel des Etablierungsprozess wurde von Problemen mit der technischen Infrastruktur und der langen Vorbereitungsphase in den Sitzungen dominiert. Diese konnten durch das Einplanen von Zeitpuffern kompensiert werden. Eine wichtige Erkenntnis in diesem Zusammenhang war, dass die Entwickler sich nur so wenig wie möglich auf die Sitzungen vorbereiteten. Später, als die technischen Probleme weitestgehend behoben waren, zeigten sich Konflikte in der Zusammenarbeit der Entwickler, weil sie sich nicht ausreichend über ihre Ziele oder Regelungen für Abläufe innerhalb der Sitzungen, z.B. der Rollenverteilung, abgestimmt hatten. Diese Konflikte konnten nur teilweise gelöst werden.
Des Weiteren führte die manchmal niedrige Priorität der DPP-Sitzungen besonders im letzten Drittel des Etablierungsprozesses zu häufigen Sitzungsausfällen. Trotzdem konnten die geplanten Projektziele erreicht werden.

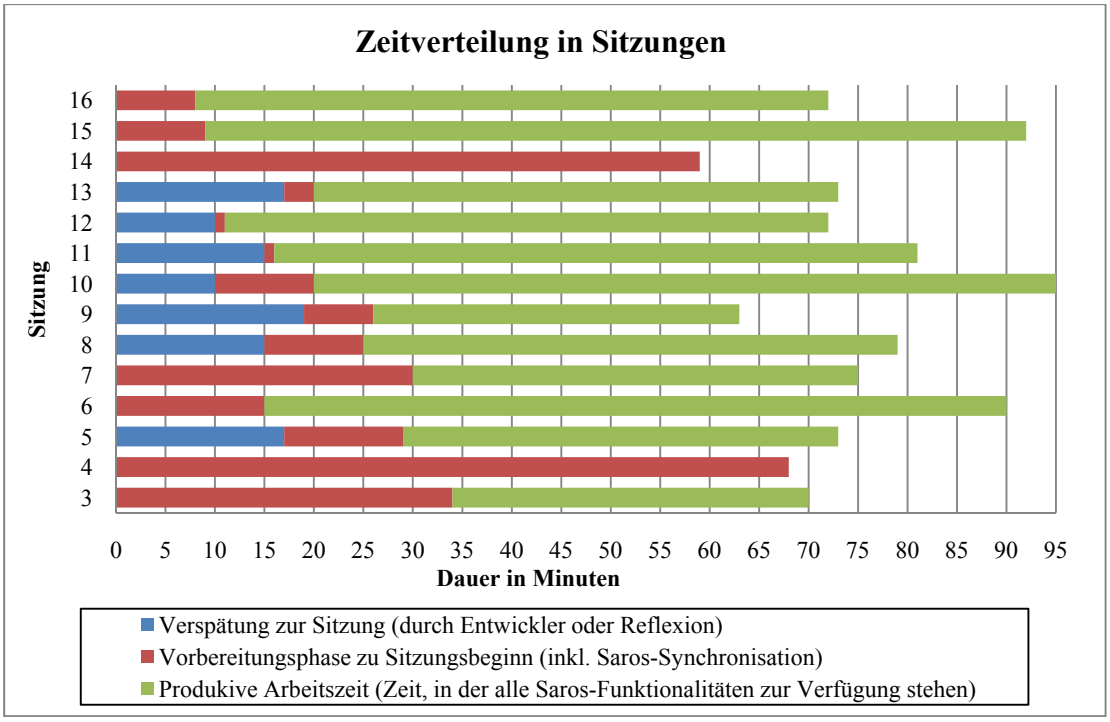


Abbildung 5.4 Zeitliche Anteile von Verzögerungen (durch Verspätungen und lange Vorbereitungsphasen) und von produktiver Arbeitszeit in den DPP-Sitzungen

5.3.2 Erkenntnisse über den Nutzen von Fragebogen

Ziele, Planung und Ablauf der schriftlichen Befragung wurden bereits in früheren Kapiteln erörtert. Nun werden der Nutzen und die Ergebnisse daraus beschrieben.
Verschiedene Fragebogentypen (GS, US, MRS) boten die Möglichkeit, allgemeine Daten über die Entwickler und konkrete Daten über die Sitzungen zu erheben. Die Ergebnisse der ersten Befragung (GQ) über demographische Daten sowie Einstellungen, Erwartungen und allgemeine Ziele der Entwickler wurden bereits in den Abschnitten 4.4.1 und 4.4.2 präsentiert.
Die Motivation der Entwickler bezüglich des Etablierungsprozesses wurde teilweise in ihrem Antwortverhalten deutlich. Die generelle Beantwortung der Fragebogen lässt auf eine Grundmotivation schließen. Wie schnell und wie ausführlich geantwortet wurde gibt einen Hinweis auf die Stärke der Motivation.
Die Informationen aus den Fragebogen (US, MRS) zu einzelnen Sitzungen waren für mich unentbehrlich, um den Ablauf und den Inhalt der Sitzungen überhaupt nachvollziehen zu können, da ich nicht mit den Entwicklern im selben Büro arbeitete und inhaltlich nicht an der Entwicklung des Codes beteiligt war. Sie brachten mir außerdem weitere Erkenntnisse über konkrete Sitzungsziele und Erwartungen der Entwickler.

Durch den Fragebogen vor der Sitzung (US) bekam ich einen Ausblick auf geplante Aufgaben und Ziele. Daraus konnte ich bereits erkennen, ob die Entwickler die gleichen Vorstellungen und Erwartungen an die Sitzung hatten. In der Sitzung konnte ich beobachten, ob die Angaben der Entwickler dem tatsächlichen Ablauf und Inhalt der Sitzung entsprachen. Der Fragebogen nach der Sitzung (MRS) zeigte mir schließlich, wie die Entwickler die Sitzung wahrgenommen hatten, ob ihre Ziele erreicht wurden bzw. ihre Erwartungen sich erfüllt hatten.
Die Erkenntnisse aus den Fragebogen kombiniert, mit meinen Beobachtungen, gaben mir einen guten Überblick über die Sitzungen. Anhand dessen konnte ich Vergleiche dazu anstellen, wie sich die Planung, der tatsächliche Ablauf und die Ergebnisse von Sitzungen zueinander verhielten, und dadurch Widersprüche aufdecken.

Erkenntnisse aus den Fragebogen

Es wurden bereits einige Angaben der Entwickler in der Beschreibung des Etablierungsprozesses (siehe Kapitel 4) und in den tieferen Analysen (siehe Abschnitt 5.2) eingebracht. An dieser Stelle werden noch weitere wichtige Angaben der Entwickler bzgl. der Erreichung der Sitzungsziele und dem Nutzen von DPP präsentiert.
Zu der Frage, ob die Entwickler die Sitzungen als Erfolg einstuften, waren sie sich immer einig. Abbildung 5.5 zeigt, was die abschließende Erhebung (siehe Abschnitt 5.1) ebenfalls noch einmal bestätigte, dass die Ziele der Entwickler in den einzelnen Sitzungen (bis auf zwei Sitzungen, die wegen technischer Probleme abgebrochen werden mussten) erreicht werden konnten.

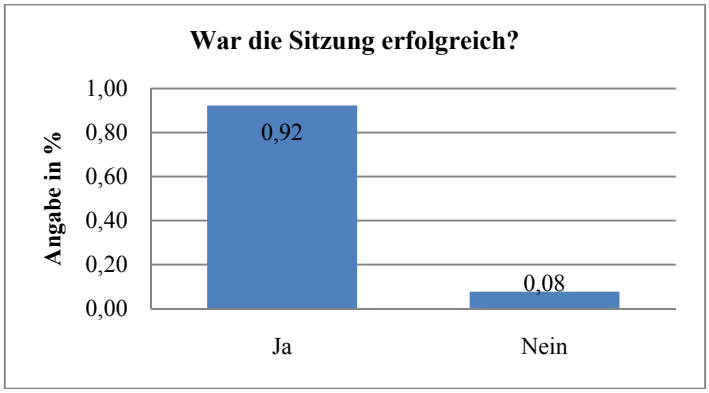
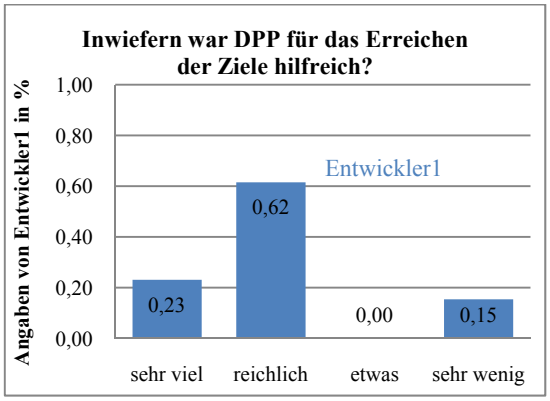
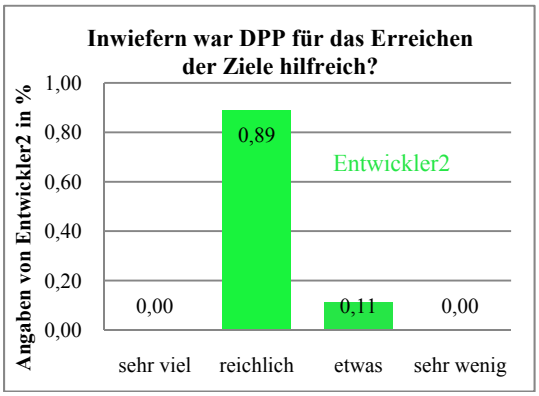


Abbildung 5.5 Angaben der Entwickler im Fragebogen nach den Sitzungen (MRS, Frage 8) über den Erfolg der Sitzung

In diesem Zusammenhang sollten die Entwickler Angaben darüber machen, wie hilfreich DPP für das Erreichen der Ziele war (siehe Abbildung 5.6). Sowohl Entwickler1 als auch Entwickler2 gaben zu über 80 % an, dass DPP hilfreich für die Erledigung der Sitzungsaufgabe war. Beide Entwickler waren einmal nicht der Meinung, dass DPP besonders hilfreich war. Entwickler1 gab in der 6. Sitzung bspw. an, dass er den Grund für eine Ausnahme (*Exception*) seiner Meinung nach ohne DPP und Erklärungen für Entwickler2 schneller gefunden hätte. In der 12. Sitzung revidierte er diese Meinung jedoch wieder und behauptete das Gegenteil (siehe Abschnitt 4.7.4).



a) Angaben von Entwickle1



b) Angaben von Entwickler2

Abbildung 5.6 a) Angaben von Entwickler1 und b) Entwickler2 im Fragebogen nach der Sitzung (MRS Frage 3) darüber wie hilfreich DPP in der Sitzung war

Anpassung der Fragebogen

Es wurde bereits bei der Erläuterung zum Ablauf des Etablierungsprozesses (siehe Abschnitt 4.6) darauf eingegangen, dass es im Zusammenhang mit Fragebogen zu Problemen kam und im Verlauf des Etablierungsprozesses Änderungen daran vorgenommen wurden.

So wurde die Beantwortung des Fragebogens MRS bspw. explizit auf das Ende der Sitzungen gelegt. Vorher hatten die Entwickler sich für die Beantwortung teilweise bis zur nächsten Sitzung eine Woche später Zeit gelassen. Durch die unmittelbare Beantwortung der Fragen konnten sich die Entwickler besser an die Geschehnisse erinnern, und mir lagen die Informationen daraus rechtzeitig für die Reflexionen vor. Außerdem bekamen die Entwickler dadurch weitere gemeinsame Zeit außerhalb der Sitzungen, die sie für spontane Fragen und ungezwungene Unterhaltungen nutzten.
Eine weitere positive Veränderung stellte sich ein, als Entwickler1 ab der 9. Sitzung von selbst anfing, eine der Fragen entsprechend der Vorgabe zu beantworten. Ich hatte bei der Vergabe von Prioritäten für Erfolgskriterien der folgenden Sitzung (US-kurz, Frage 7) darum gebeten, jede Priorität nur einmal zu vergeben, um eine Abstufung zwischen den erfolgsrelevanten Faktoren zu erhalten. Die Entwickler hatten jedoch von Anfang an Prioritäten mehrfach vergeben.
Im deskriptiven Teil dieser Arbeit (siehe Kapitel 4) wurde ebenfalls erwähnt, dass die Fragebogen im Verlauf des Etablierungsprozesses gekürzt wurden, weil die Entwickler die Beantwortung als zu großen Aufwand empfanden. Gekürzt wurden z.B. Kommentarfelder, die sich als wenig ergiebig erwiesen hatten. Gleichzeitig wurden Fragen, die ähnliche Ergebnisse lieferten, zusammengefasst (siehe lange und kurze Versionen von US und MRS in Anhang B).

Probleme im Zusammenhang mit Angaben im Fragebogen

Um ein Urteil darüber abgeben zu können, ob die Sitzungen nutzbringend waren, war es nötig, zwischen den Angaben der einzelnen Entwickler und den Ereignissen in der Sitzung zu differenzieren. Es gab bspw. Widersprüche sowohl zwischen den Angaben der einzelnen Entwickler als auch zwischen den Angaben der Entwickler und den tatsächlichen Geschehnissen in der Sitzung.
Entwickler1 hatte bspw. in den Sitzungen 12 und 13 angegeben, dass er sich nicht auf die Sitzungen vorbereitet hatte. Zu Sitzungsbeginn wusste er jedoch bereits an welcher Stelle Änderungen am fremden Code vorgenommen werden müssten und hatte dies vor den Sitzungen jeweils mit dem ursprünglichen Autor besprochen. Ein weiteres Beispiel für Angaben der Entwickler, die sich nicht mit dem Inhalt der Sitzungen deckte, findet sich ebenfalls in Sitzung 13: Entwickler2 gab an, dass die Aufgaben der Sitzung wie erwartet durchgeführt werden konnten, obwohl die erste Aufgabe abgebrochen werden musste.
In der Planungsphase hatte ich befürchtet, dass es einige Fragen geben könnte, deren Beantwortung durch Konformitätsdruck beeinflusst werden würde. Dies zeigte sich sowohl in den Sitzungen als auch in Fragebogen. Beispielsweise gab Entwickler2 auf die Frage nach seinen Präferenzen für die Herangehensweise an neue Aufgaben im ersten Fragebogen (GQ, Frage 3) an: „I don't have any problem with anything." Zudem ließ er sich auch durch Nachfragen nicht dazu überreden, die Prioritäten für eine der Fragen (US-kurz, Frage 7) nur einmal zu vergeben, sondern bezeichnete alle Faktoren als 1= besonders wichtig oder 2=wichtig, anstatt die sechs Faktoren von eins bis sechs nach ihrer Wichtigkeit zu priorisieren.

Außerdem befürchtete ich, dass offene Fragen Antworten auf einer unerwarteten Abstraktionsebene erbringen könnten. Dies geschah bspw. im ersten Fragebogen (GQ, Frage 5), als Entwickler2 auf die Frage nach seiner allgemeinen Meinung über DPP mit einer Beschreibung der Praktik antwortete. Insgesamt unterschied sich der Detailgrad der Antworten der Entwickler stark und machte es teilweise schwierig sie miteinander zu vergleichen.

Auch geschlossene Fragen lieferten nur teilweise verständliche Antworten, denn bei der Beantwortung der Frage ob Aufgaben wie erwartet in den Sitzungen bearbeitet werden konnten (MRS, Frage 5) lieferte die Auswahl (wie erwartet, schneller oder langsamer) ohne Begründung nur un-

zureichende Informationen. Die Reflexionen boten dann die Gelegenheit, genauere Informationen von den Entwicklern zu erfragen.

5.3.3 Erkenntnisse über den Nutzen von Reflexionen

In der Beschreibung des Sitzungsablaufes wurde bereits erwähnt, dass die Reflexionen zur Besprechung von optimierenden Maßnahmen anhand der bereits gesammelten Erkenntnisse dienten. Beispielsweise wurde darin die Planung des Etablierungsprozesses wie z.B. Sitzungsdauer und Termine besprochen (siehe Abschnitt 4.6). Erörterungen über Versuche, die lange Vorbereitungsphase in den Sitzungen zu vermeiden, war auch immer wieder Thema von Reflexionen.
Über die technische Infrastruktur wurde in den Reflexionen gesprochen, wenn Einstellungsänderungen anstanden, oder wenn der Hergang von technischen Problemen aus der Sitzung rekonstruiert wurde. Zudem wurden Anforderungen an Saros erörtert.
Weiterhin konnte ich mir Beobachtungen aus den Sitzungen bestätigen lassen oder Angaben in Fragebogen hinterfragen, z.B. bei unklaren Angaben oder Angaben ohne Erklärungen.
Die Reflexionen dienten den Entwicklern darüber hinaus als zusätzliche gemeinsame Zeit außerhalb der Sitzungen, um sich besser kennenzulernen, da diese sich vor Beginn der Sitzungen kaum bekannt waren.
Wie schon im Abschnitt zur Zielsetzung und Planung der Reflexion (siehe Abschnitt 3.2.2) erwähnt, sollten Reflexionen jedoch vor allem dazu genutzt werden, während des DPP-Projektes Erfahrungen über den Etablierungsprozess auszutauschen und somit eine Rückkopplung der Erkenntnisse über diesen Prozess zu ermöglichen.
Die Kommunikation innerhalb der Reflexionen war allerdings stockend. Die Entwickler tauschten sich nicht direkt aus. Auf Fragen und Themenvorschläge meinerseits, z.B. wie mit der Rollenverteilung in Sitzungen umgegangen werden soll, gingen sie meist nur kurz und oberflächlich ein. Diese Problematik wird in den folgenden Absätzen zu Anpassung und Problemen ausführlicher behandelt.

Anpassung der Reflexionen

Manche Anpassungen zu Reflexionen wurden ebenfalls in Abschnitt 4.6 und 5.2.3 schon erwähnt. Zur besseren Überschaubarkeit der besprochenen Themen in Reflexionen wurde ab der vierten Reflexion (zu Sitzung 7) Saros zur visuellen Unterstützung eingeführt. Besprochene Themen wurden in Stichpunkten für alle Beteiligten sichtbar gemacht.
Um zusätzliche Maßnahmen des Etablierungsprozesses für die Entwickler so wenig aufwendig wie möglich zu gestalten, wurden zwei weitere Anpassungen von Reflexionen durchgeführt. Erstens wurden Reflexionen an Sitzungstagen geplant, damit die Entwickler für beides nur einen gemeinsamen Termin finden mussten. Wegen Verspätungen des Sitzungsbeginns durch Reflexionen wurden sie nur noch im Anschluss an Sitzungen gelegt. Zweitens wurden Reflexionen nicht mehr für einzelne Sitzungen sondern für alle Sitzungen einer Woche durchgeführt.
Ab dem dritten Arbeitszyklus wurde den Entwicklern schrittweise die Verantwortung für die Reflexionen überlassen. Die Gründe hierfür werden im nächsten Absatz erläutert.

Probleme mit Reflexionen

Von Anfang an bestand das Problem, dass die Entwickler in Reflexionen nicht direkt miteinander sprachen. Sie verhielten sich passiv und antworteten nur auf meine Fragen, anstatt von sich aus etwas zum Gespräch beizutragen. Dies ließ sich anfangs dadurch begründen, dass die Entwickler angenommen hatten, die Reflexionen sollten nur ein zusätzliches Interview meinerseits sein. Daher erklärte ich noch einmal, welche Funktion die Reflexionen im Etablierungsprozess haben sollten.

Die Reflexionen wurden also erst einmal in Form eines Interviews mit von mir ausgewählten Themen weitergeführt, um die Geschehnisse im Etablierungsprozess überhaupt zur Sprache zu bringen.
Das nächste Problem, das sich stellte, war die Tatsache, dass die Entwickler sich nicht auf das Gespräch konzentrierten, sich gegenseitig nicht zuhörten und sich nicht besonders involvierten. Ich fühlte mich teilweise an Schulzeiten erinnert, wenn einer der Entwickler so knapp und allgemein wie möglich antwortete, während die anderen im Hintergrund tuschelten und lachten, ohne dem Gespräch in der Reflexion zu folgen. Entwickler2 äußerte mir gegenüber Zweifel an der Notwendigkeit, außerhalb der Sitzungen miteinander zu reden.
Daraufhin versuchte ich vor allem Themen anzusprechen, die den konkreten Ablauf der Sitzungen betrafen. Dies waren Themen, bei denen die Entwickler das Bedürfnis bekommen sollten, miteinander zu reden, Erfahrungen auszutauschen und ein gemeinsames (Problem-)Verständnis zu entwickeln wie z.B. die Abstimmung von Sitzungszielen oder der Umgang mit aktuellen Konflikten.
Zur Verbesserung der Atmosphäre in den Reflexionen suchte ich mir zudem Rat bei einer Dozentin des Fachbereichs Psychologie, die sich auch freiberuflich auf Personalentwicklung spezialisiert hatte. In der gemeinsamen Analyse der Reflexionen stellten wir fest, dass die Situation vermutlich auch durch Probleme in der Teamzusammenstellung oder durch Teamentwicklungskonflikte verursacht worden war.
Weiterhin fühlten die Entwickler sich nicht verantwortlich oder nicht im Stande, sich aktiv an den Sitzungen zu beteiligen. Sie wirkten teilweise unkonzentriert und hörten sich gegenseitig nicht zu. Es fiel ihnen schwer, sich gegenseitig oder den Prozess zu bewerten.
Dies änderte sich erst in der Reflexion zur 13. Sitzung, in der probeweise Entwickler1 die Gesprächsführung übertragen bekam. Zur Vorbereitung hatte ich mit ihm verschiedene Themen in einem kurzen Einzelgespräch, das sich durch eine weitere Verspätung von Entwickler2 ergeben hatte, erörtert.
In der Reflexion sprach er jedoch nur die Verspätung an und beendete das Gespräch sofort danach. Zum ersten Mal hatten die Entwickler in einer Reflexion aktiv eine Rolle übernommen und sich direkt miteinander ausgetauscht.
Die Übertragung der Reflexionsmoderation an einen der Entwickler brachte jedoch zwei neue Probleme auf. Zum einen blieben durch das abrupte Ende noch Gesprächsthemen offen. Zum anderen war die Reflexion unausgewogen, weil einer der Entwickler auf die Themen vorbereitet war und mehr Verantwortung hatte als der andere.
Daher wurden für die nächste Reflexion beide Entwickler vorher per Email über Themen und den Ablauf (ähnlich des Reflexionsleitfadens in Anhang C) informiert. Dieses Mal wurde Entwickler2 die Moderation der Reflexion übertragen. Des Weiteren nahm ich mir vor, am Ende der Reflexion selbst noch einmal Fragen zu stellen, um offengebliebene Punkte anzusprechen.
Auch diese Variante bewirkte, dass die Entwickler sich von selbst über die Geschehnisse in der Sitzung besprachen. Entwickler2 fühlte sich geschmeichelt, dass er die Verantwortung für die Reflexionsleitung übertragen bekommen hatte. Er vermied es jedoch geschickt, über seine eigenen Erfahrungen zu sprechen, indem er die Reflexion wie ein Interview mit Entwickler1 gestaltete.
Es zeigte sich immer wieder, dass es den Entwicklern, bis auf wenige Ausnahmen, äußerst schwer fiel, gegenseitig Kritik zu äußern. Beispielsweise wartete Entwickler1 bis zur 13. Sitzung, um Entwickler2 darauf hinzuweisen, nicht zu spät zu den Sitzungen zu kommen.
Wie in Abschnitt 4.6 bereits erwähnt führte die letzte Anpassung dazu, dass eine spontane Absage der letzten Sitzung einer Woche auch dazu führte, dass für diese Woche die Reflexion ausfiel.
Weiterhin war es auch problematisch Reflexionen durchzuführen, wenn sie nicht sehr zeitnah zu den Sitzungen stattfanden. Meistens konnten die Entwickler sich schon am nächsten Tag nicht mehr genau an Geschehnisse aus der Sitzung erinnern. Das Einzige was gegen zeitnahe Reflexionen sprach war, dass ich dadurch wenig Zeit zur Vorbereitung hatte und die Fragebogen bis dahin oftmals noch nicht alle vorlagen.
Auch die Befürchtung aus der Planungsphase, dass Entwickler sich evtl. nicht in Gruppengesprächen einbringen konnten, bewahrheitete sich. Beispielsweise fragte ich in einer Reflexion, die zu viert durchgeführt wurde, ob Entwickler2 auch noch etwas sagen wollte, weil bisher hauptsäch-

lich die anderen beiden Entwickler gesprochen hatten. Er fing seine Antwort mit: *„Yes, I also wanted to say something …“* an. Er wartete also bis ich ihn ansprach, statt mir zu signalisieren, dass er etwas zu sagen hatte.
Was am Ende noch fehlte war, dass die Entwickler eigene Beiträge in die Reflexionen einbrachten. Die Gespräche in Reflexionen basierten bis zum Schluss auf Themen, die von mir ausgewählt wurden. Trotzdem konnten die Reflexionen genutzt werden, um Erkenntnisse in den Etablierungsprozess einfließen zu lassen und neue Informationen zu sammeln.

5.3.4 Weitere Erkenntnisse aus dem Etablierungsprozess

Mehrmals war es notwendig, im Etablierungsprozess den geeigneten Typ der Befragung bzw. Kommunikationsform neu festzulegen. Da anfangs noch zu wenige Informationen über das Projekt und die Beteiligten vorlagen, eigneten sich Fragebogen mit offenen Fragen am besten zur Datenerhebung. Für erste Erörterungen über Geschehnisse aus den Sitzungen eigneten sich Gruppendiskussionen innerhalb der Reflexionen, durch die die Entwickler auch ein gegenseitiges Verständnis füreinander entwickelten. Für das Ergreifen von konkreten Maßnahmen, die mit der Erweiterung des Etablierungsprozesses zusammenhingen, boten sich auch Einzelgespräche mit Entwickler1 an, da er hauptsächlich die Entscheidungen über den Entwicklungsprozess traf. Am Ende wurden im Rahmen der abschließenden Erhebung Einzelgespräche geführt. Hauptsächlich sollten die Entwickler die Möglichkeit bekommen, sich unabhängig und unbeeinflusst voneinander über ihre Erfahrungen zu äußern. Besonders bemerkenswert war die Tatsache, dass mündliche Befragungen in manchen Fällen dazu führten, dass die Entwickler Rückfragen stellten, mit denen ich nicht gerechnet hatte. Beispielsweise als Entwickler2 in einer der Reflexionen in Frage stellte, dass überhaupt außerhalb der Sitzungen kommuniziert werden müsste.

Erkenntnisse über den Ablauf des Etablierungsprozesses wurden bereits in Abschnitt 4.6 teilweise erläutert – so bspw. die Tatsache, dass die Festlegung der Sitzungstage vor allem von den geplanten Aufgaben und der Verfügbarkeit der Entwickler abhing, oder dass die Vereinbarung der Sitzungszeiten durch die verschiedenen Zeitzonen beeinflusst wurde.
Erkenntnisse über Fragebogen und Reflexionen befinden sich in den Abschnitten 5.3.2 und 5.3.3.
Die ausführliche Vorbereitungsphase inkl. der Zusammenstellung der technischen Infrastruktur und der Einführungsmaßnahmen in den Pilotsitzungen stellte sich als sehr hilfreich heraus. Dadurch waren bereits vor Sitzungsbeginn alle technischen Voraussetzungen geklärt, die Entwickler hatten bereits erste Erfahrungen im Umgang mit der technischen Infrastruktur gesammelt, und es wurden schon einmal alle Maßnahmen und Ziele des Etablierungsprozesses besprochen, wodurch in den folgenden Sitzungen Zeit gespart werden konnte. Weiterhin ließen sich dadurch bereits erste technische Mängel identifizieren und das Aufzeichnungsverfahren im praktischen Einsatz testen.
Es zeigte sich, dass mindestens eine Sitzung pro Woche durchgeführt werden musste, um die Kontinuität der Sitzungsinhalte und die Routine im Umgang mit der technischen Infrastruktur zu gewährleisten.

Der Nutzen der Betreuung und der weiteren Maßnahmen im Etablierungsprozess zeigte sich schon bei den Vorbereitungen des Projektes, bei der Bereitstellung der technischen Infrastruktur und in der Pilotphase. Die Betreuung hätte vielleicht wirkungsvoller sein können, wenn ich die Entwickler aus Bangalore vorher persönlich kennengelernt hätte, denn bis zuletzt fiel es diesen Entwicklern schwer, die Betreuung als Hilfe und nicht als Kontrolle zu verstehen. Während des Etablierungsprozesses war die Betreuung zum einen sinnvoll, um sowohl technische als auch andere Probleme aufzudecken und zu behandeln. Zum anderen stellte sich heraus, dass ohne Betreuung keine Sitzungen zustande kamen.

Es wurden im Besonderen zwei Grenzen aufgezeigt. Zum einen konnten Funktionalitäten, z.B. hinsichtlich der Awareness oder des Multi-Driver-Modes, zwar bereitgestellt und erklärt werden. Dies führte jedoch nicht unbedingt dazu, dass sie in den Sitzungen auch genutzt wurden. Zum

anderen hatte es wenig Auswirkungen die Entwickler auf Probleme wie z.B. schlechte Vorbereitung oder die Konflikte mit Rollenwechseln hinzuweisen, wenn ihre Bereitschaft zu einer Aussprache oder Änderungen ihres Verhaltens fehlte.

5.3.5 Erkenntnisse über erweiternde Maßnahmen im Etablierungsprozesses

In diesem Abschnitt wird erläutert, wie sich der Etablierungsprozess in Hinsicht auf die Erweiterung des Einsatzgebietes und der Teilnehmer entwickelt hat. Die möglichen Erweiterungen im Etablierungsprozess hingen weitestgehend von der Einschätzung von Entwickler1 ab. Dabei spielten zwei Faktoren eine Rolle, zum einen ob die technische Infrastruktur zuverlässig genug funktionierte und zum anderen ob die Erweiterungen in seine Planung integrierbar waren (siehe Beschreibung zu Sitzung 5).

Die Sitzungstypen wurden von Entwickler1, abhängig von geplanten Sitzungsinhalten, beteiligten Entwicklern und dem Status des Entwicklungsprozesses, gewählt. Es kamen zwei verschiedene Sitzungstypen in den Sitzungen vor: Pair-Developing und Review. Die Review-Sitzungen entsprachen Walkthroughs, bei denen die Entwickler hauptsächlich die semantische Korrektheit untersuchten mit dem Ziel, den Code kennenzulernen und die Umsetzung eines Proof of Concept zu prüfen. Die Pair-Developing-Sitzungen enthielten mehrmals Einführungen in bereits existierenden Code und verschiedene andere Subtypen, z.B. Planung oder Diskussionen über Anforderungen. Ein weiterer Sitzungstyp, Debugging, war geplant, kam jedoch nicht zum Einsatz, weil die betreffende Sitzung wegen technischer Probleme mit Eclipse abgebrochen werden musste.
Mehrmals äußerten die Entwickler sich in den Fragebogen und in Reflexionen zu ihren Erfahrungen in Pair-Developing-Sitzungen. Entwickler1 und Entwickler2 waren sich darin einig, dass grundsätzlich die angestrebten Ziele in den Pair-Developing-Sitzungen immer erreicht werden konnten (siehe abschließende Erhebung). Sie äußerten sich jedoch anfangs kritisch, wenn es darin zu Problembehandlungen kam. In diesen Fällen hätten sie lieber alleine weitergearbeitet. Entwickler2 gab in diesem Zusammenhang an, dass er sich besonders bei denkintensiven Aufgaben durch seinen Partner unter Druck gesetzt fühlte und sich daher solche Aufgaben nicht für DPP eigneten. Entwickler1 gab zu Sitzung 6 an, dass er Ursachen für Defekte lieber alleine gesucht hätte, weil es ohne Kommentierung für den anderen Entwickler seiner Meinung nach schneller gegangen wäre. In der 13. Sitzung behauptete er allerdings genau das Gegenteil, indem er sagte, dass gerade durch die Zusammenarbeit in der Sitzung Problemlösungen gemeinsam sehr schnell gefunden werden konnten.
Alle Entwickler waren sich darin einig, dass sich auch Reviews, genauer gesagt Walkthrougs, als Sitzungstyp für DPP-Sitzungen eignen.

Die Anzahl der möglichen beteiligten Entwickler wurde dadurch beschränkt, dass es sich bei dem Projekt um ein Java-Projekt der C5-Abteilung handelte. Das Projekt wurde in Wien von Entwickler1 bearbeitet, in Bangalore waren Entwickler2 und Entwickler3 die einzigen Java-Entwickler.
Unter diesen Beteiligten wurde von Entwickler1 bestimmt, wer an Sitzungen teilnehmen sollte. Die Sitzungen fanden fast ausnahmslos mit zwei Entwicklern statt. Mehrere Versuche eine Sitzung mit einem dritten Teilnehmer zu etablieren, scheiterten daran, dass dem dritten Entwickler keine sinnvolle Rolle zugeordnet werden konnte (siehe Abschnitt 5.1) und deswegen der dritte Entwickler der Bearbeitung anderer Projekte eine höhere Priorität einräumte.

Auch der Versuch, eine weitere Funktionalität von Saros, die parallele Mehrbenutzer-Unterstützung (Multi-Driver-Mode), einzuführen scheiterte daran, dass die Entwickler in ihren Sitzungen keine Verwendung dafür hatten. Die Art mit der Entwickler1 sein Wissen vermittelte erforderte, dass die Entwickler aus Bangalore seine Aktivitäten immer unmittelbar verfolgten.

5.3.6 Der Einfluss des Etablierungsprozesses auf die Entwicklung von Saros

Im Abschnitt über Saros wurde bereits erklärt, dass es sich bei dieser Software um eine Open Source Software (kurz OSS) handelt. Bei OSS-Projekten haben die Benutzer und die Entwickler oft engen Kontakt zueinander. Somit haben die Benutzer relativ viel Einfluss auf die Weiterentwicklung. Zudem bekommen die Entwickler direktes Feedback über die Tauglichkeit und Benutzerfreundlichkeit von Funktionalitäten und Fehlfunktionen ihrer Software.
Benutzer können meist über verschiedene Kommunikationskanäle (Bugtracker, Foren, Mailinglisten, E-Mail, etc.) mit den Entwicklern in Verbindung treten oder unter bestimmten Bedingungen auch über Quellcodebeiträge eigene Funktionalitäten zum Projekt beisteuern.
Durch das DPP-Projekt mit Teles konnten zusätzlich zu den üblichen Kommunikationswegen weitere Informationsquellen genutzt werden, um Feedback über die Einsatzfähigkeit von Saros zu erhalten. Im Rahmen der DPP-Sitzungen kamen folgende Kommunikationswege hinzu: Beobachtungen in den Sitzungen, Saros-Error-Logs (dt. Fehlerberichte) der einzelnen Entwickler aus den Sitzungen und direktes (mündliches und schriftliches) Feedback der Teles-Entwickler.
Die Entwickler von Teles hatten nicht nur durch ihr Feedback eine besondere Stellung zum Saros-Projekt. Im Rahmen der technischen Betreuung der Sitzungen wurden Änderungen, die sich aus den Sitzungen ergaben, soweit die Ressourcen es zuließen, bei der Weiterentwicklung von Saros mit relativ hoher Priorität berücksichtigt. Meistens reagierte das Saros-Team bereits unmittelbar nach der Sitzung auf alle technischen Probleme im Workspace, die sich nicht sofort lösen ließen, mit einem konkreten Vorschlag für eine vorläufige alternative Vorgehensweise.
Die großen Abstände zwischen den Sitzungen verlangsamten einerseits den Fortschritt des DPP-Projektes. Sie kamen der Weiterentwicklung von Saros jedoch andererseits auch entgegen. Die Zeiträume zwischen den Sitzungen ermöglichten genauere Analysen technischer Vorkommnisse und die Suche nach alternativen Vorgehensweisen. Gab es für ein Problem keine unmittelbare alternative Vorgehensweise, so wurde die Bearbeitung an einen der Entwickler von Saros übertragen und meist schon in das nächste Release wenige Wochen später übernommen. Beispielsweise wurde eine zweite Variante der Projektsynchronisation mit einer gezielten Auswahl von Dateien nur für die Bedürfnisse von Teles in Saros eingebaut. Weitere Einflüsse, die Teles auf Saros hatte, werden in den folgenden Abschnitten beschrieben.

Arten und Ursprung von Feedback

Das Feedback, das sich aus Sitzungen mit Teles ergab, betraf nicht nur die technische Ebene, sondern bezog sich auch auf Awareness oder andere Aspekte der Benutzerfreundlichkeit. Es wurde wie oben erwähnt entweder implizit (durch Beobachtungen des Saros-Teams) oder explizit (durch Angaben der Entwickler, z.B. in Fragebogen) erhoben.
Neue Anforderungen und Fehlermeldungen wurden teilweise in den Fragebogen zur Sitzung oder in der Sitzung mitgeteilt. Beispielsweise teilte Entwickler1 mit, dass er Schwierigkeiten mit seinem Dateiverwaltungssystem hätte, weil Saros die Zeilentrennung in den synchronisierten Dateien verändern würde. Manche Defekte konnten durch Beobachtungen in den Sitzungen festgestellt werden, so bspw. das Problem, dass die Synchronisation nach wenigen Sekunden automatisch abgebrochen wurde, oder dass der Verfolgermodus vom Observer nicht aktiviert werden konnte.
Für Rückfragen zum Feedback der Entwickler oder zu Einstellungsänderungen konnten die Reflexionen genutzt werden.
Durch Analysen der aufgezeichneten Sitzungen konnten weitere Anforderungen ermittelt werden. Beispielsweise stellte der Leiter der Saros-Entwicklung fest, dass es sinnvoll wäre, wenn der Synchronisationsprozess auch im Hintergrund laufen könnte, sodass die Entwickler in der Zwischenzeit Zugriff auf andere Daten in Eclipse hätten.
Aus den Saros-Error-Logs, die die Teles-Entwickler nach den Sitzungen übermittelten, konnten zusätzliche Informationen zu fehlerhaften technischen Vorgängen, z.B. Fehler beim Ablauf von Aktivitäten, durch Fehlermeldungen rekonstruiert und später behoben werden.

Neuerungen durch Teles

Im Verlauf des Projektes ergaben sich zwei allgemeine Vorgehensweisen: (1) wie man mit neuen Anforderungen umgehen kann und (2) wie man mit technischen Problemen in der Sitzung umgehen kann. Neue Anforderungen wurden von mir in einem zentralen Dokument hinterlegt. Danach wurden die Verantwortlichen informiert. Sie entschieden über den Umgang mit den neuen Anforderungen und hinterlegten dies in Form von Kommentaren zu den Anforderungen.
Um den Umgang mit technischen Problemen in Sitzungen so effektiv wie möglich zu gestalten, wurde ein Vorgehen für das Saros-Team eingeführt. Dabei lag es im Ermessen der Beteiligten (vor allem der Betreuung), die Sitzung nach einer bestimmten Dauer und Anzahl von Maßnahmen abzubrechen und die Problembehandlung auf einen geeigneten Zeitpunkt außerhalb der Sitzungen zu verschieben.
Es stellte sich weiterhin im Verlauf des DPP-Projektes heraus, dass das Error-Log für die Rekonstruktion von Problemen und die Lokalisierung von Problemquellen aus den Sitzungen essentiell war. Es wurde dafür sogar noch um einige Elemente wie z.B. die verwendete Java-Version mit der Eclipse betrieben wurde, erweitert. Da das Saros-Team darauf angewiesen war, die Log-Dateien von den Entwicklern einzusammeln, sollten zur Vereinfachung im Umgang alle Dateien eines Tages in einem gemeinsamen Ordner verwaltet werden. In einem späteren Release konnte eingerichtet werden, dass sie auch automatisch an das Saros-Team verschickt wurden.
Ein weiterer Einfluss auf Saros war der Versuch, in einer der Sitzungen einen neuen Sitzungstyp (Debugging) einzuführen, der bisher von Saros nicht unterstützt wurde. Dafür wurde vor der Sitzung von mir ein Anwendungsfall konstruiert und überprüft, inwiefern Saros die gewünschten Vorgänge bereits unterstützte. Des Weiteren folgten Besprechungen mit dem Leiter der Saros-Entwicklung über mögliche Anpassungen von Saros. Da diese Sitzung wegen technischer Probleme ausfiel, das Saros-Team für Debugging-Sitzungen den Einsatz von Desktop Sharing empfahl und kein weiterer Bedarf für diesen Sitzungstyp vorhanden war, wurden in dieser Hinsicht keine weiteren Erörterungen angestellt.
Als feststand, dass der dritte Entwickler an den Sitzungen teilnehmen würde, stellte sich die Frage ob es für einen Benutzer möglich sein würde, eine ältere Version von Saros zu installieren. Bis dahin war immer nur die aktuellste Version auf der Website verfügbar, doch die anderen beiden Entwickler verwendeten eine ältere Version, mit der die neueste Version, nicht kompatibel war. Dieses Problem löste sich jedoch von selbst, da alle Entwickler mit der neuesten Version weiterarbeiteten.
Es war mehrfach in Sitzungen vorgekommen, dass technische Probleme auftraten, weil die Entwickler verschiedene Versionen von Saros installiert hatten. Um Kompatibilitätskonflikte zwischen Versionen schneller als Problemquelle ausschließen zu können, wurde eine Prüfung der Versionen am Anfang der Sitzung in Saros eingebaut.
Saros befand sich noch in der α-Version der Entwicklung als das Projekt begann. Die Software verursachte besonders anfangs viele technische Probleme, und auch später zeigten sich noch einige Stellen, an denen sie optimiert werden konnte. Es stellte sich im Nachhinein heraus, dass es sinnvoll war, die technische von der prozessualen Betreuung zu trennen, und sie von jemandem durchführen zu lassen, der an der Entwicklung beteiligt war und den Code sowie das erwartete Verhalten gut kannte.

Unerfüllte Anforderungen

Manche Anforderungen von Teles blieben während des DPP-Projektes unerfüllt, weil sie entweder nur wenig Relevanz für die Sitzung hatten (wie z.B. das Debugging) oder weil nach Abwägung durch das Saros-Team festgestellt wurde, dass sich für die Betrachtung von manchen Artefakten andere Anwendungen besser eigneten. Beispielsweise äußerten die Entwickler den Wunsch, gemeinsam Artefakte wie z.B. Testberichte und Webinterfaces zu betrachten, wofür sich Desktop Sharing-Anwendungen in den meisten Fällen besser eignen würden.

5.4 Allgemeine Anleitung für einen optimierten DPP-Etablierungsprozess

Die folgende Anleitung entspringt den Erfahrungen aus dem Etablierungsprozess. Es ist eine Zusammenfassung von Maßnahmen, die dabei hilfreich waren und Vorschlägen dazu wie aufgetretene Probleme zukünftig evtl. vermieden werden können. Sie ist nicht vollständig, sondern soll eine Orientierungshilfe und praktische Unterstützung für die Betreuung von DPP-Etablierungsprozessen bieten.

Voraussetzungen für den DPP-Etablierungsprozess schaffen

1. Zuerst sind alle notwendigen technischen Voraussetzungen zu schaffen, wie sie bereits in der Planungsphase und bei der technischen Infrastruktur beschrieben wurden (siehe Abschnitte 4.2 und 4.3).
    a. Die geeignete Software entsprechend den Anforderungen, z.B. Audio bzw. Videoverbindung und Groupware, wählen. Dabei zeigte sich, dass es für Beobachtungen sinnvoller ist, ein Desktop Sharing zu einem der Entwickler aufzubauen, um auch Informationen über Aktivitäten außerhalb des Workspace zu erhalten. Beispielsweise zeigte sich auch der Bedarf einer zweiten Netviewer-Verbindung bzw. Videoübertragung, wenn das Feedback unzureichend war oder Entwickler abgelenkt wirkten.
    b. Die geeignete Hardware, z.B. USB-Headsets, Bildschirme mit einer bestimmten Auflösung, etc., wählen.
    c. Die technischen Kapazitäten entsprechend der geplanten Infrastruktur sicherstellen: Internetverbindung mit ausreichender Bandbreite und wenig Latenz sowie Rechner mit ausreichender Leistung.
    d. Einen praktischen Testlauf mit allen beteiligten Entwicklern durchführen, wodurch sie auch erste Erfahrungen und Übung im Umgang mit der technischen Infrastruktur erlangen.
    e. Technische und inhaltliche Betreuung trennen, wenn die technische Infrastruktur nicht problemfrei funktioniert. Dies gibt dem Betreuer die Möglichkeit, sich entweder auf technische oder auf nicht technische Probleme bei der Betreuung der Sitzung zu konzentrieren.
2. Weiterhin sind alle zusätzlichen Maßnahmen des Etablierungsprozesses soweit wie möglich vorzubereiten.
    a. Allgemeine Befragungsformen festlegen und vorher von Testpersonen prüfen lassen.
    b. Das Aufzeichnungssetting zur Datensicherung festlegen und testen, damit keine wichtigen Daten verloren gehen, die evtl. für spätere Analysen interessant sind.
    c. Die technische Infrastruktur so stabil und zuverlässig wie möglich wählen, denn technische Probleme verhindern Fortschritte im Etablierungsprozess.
3. In einem Abkommen zum Ablauf des Etablierungsprozesses sind zwischen dem Betreuer und den Entwicklern die Regeln für den Etablierungsprozess festzulegen.
    a. Die Ziele der Entwickler und Maßnahmen des Betreuers besprechen und damit eine gemeinsame Basis für die Zusammenarbeit schaffen. Es ist anzunehmen, dass ein persönliches Treffen in diesem Zusammenhang zu mehr Vertrauen führt.
    b. Gemeinsam Teilergebnisse und die Dauer der Zyklen planen, damit die Entwickler regelmäßig über Forschungsergebnisse auf dem Laufenden gehalten werden, Statusprüfungen während des Etablierungsprozesses auf dieser Basis möglich sind. Für den Betreuer wird der Etablierungsprozess durch die Einteilung in Zyklen überschaubarer.
    c. Die Ergebnisse der einzelnen Zyklen mit der Planung abgleichen und bei jedem Zyklus die Teilziele ergänzen.

Ablauf und Organisation von Sitzungen im Etablierungsprozess festlegen

4. Es sind konkrete Regeln aufzustellen, die die Kontinuität des Etablierungsprozesses sicherstellen.
    a. Regelmäßige Sitzungen stattfinden lassen. Es eignet sich bereits eine Sitzung pro Woche mit einer Dauer von 90 Minuten. In der Dauer einen Puffer von 15 Minuten

für evtl. auftretende Probleme, z.B. mit der technischen Infrastruktur oder wegen Fehleinschätzungen des Aufwandes von Aufgaben, einplanen.

b. Durch die Planung sicherstellen, dass alle Entwickler regelmäßig an Sitzungen teilnehmen, denn sonst verlieren die Entwickler den Anschluss an den Etablierungsprozess und finden sich selbst in den Sitzungen, z.B. mit der technischen Infrastruktur, nicht gut zurecht.

c. Den Nutzen der Maßnahmen, z.B. Reflexionen, regelmäßig prüfen. Beispielsweise die Wahl der Befragungsform. Gruppengespräche eignen sich zur gemeinsamen Problemerforschung und Entwicklung eines gemeinsamen Verständnisses. Ein Einzelgespräch ist hingegen sinnvoll, wenn Beiträge der Entwickler unabhängig und unbeeinflusst erfolgen sollen oder die Maßnahme nur einen der Entwickler erfordert, z.B. die Ermittlung des Standardentwicklungsprozesses.

d. Befragungen, z.B. Fragebogen und Gespräche, so zeitnah wie möglich zu Sitzungen durchführen, damit die Entwickler sich noch gut erinnern können und der Betreuer die Daten so früh wie möglich zur Verfügung hat.

e. Die Kommunikation zwischen allen Beteiligten des Etablierungsprozesses transparent gestalten, d.h. bspw. Emails immer an alle Beteiligten verschicken. Zur Unterstützung der Kommunikation wird empfohlen ebenfalls eine zentrale Plattform, z.B. ein Wiki, zu verwenden, um Termine, Maßnahmen sowie weitere Informationen für alle Beteiligten jederzeit zugänglich zu machen.

5. Vor jeder Sitzung sind die nötigen Voraussetzungen sicherzustellen.

   a. Ziele, Aufgaben und die nötige Vorbereitung für die Sitzungen festlegen und zwischen allen Beteiligten abstimmen.

   b. Durch Abfragen am Anfang der Sitzung sicherstellen, dass die vereinbarte Vorbereitung erfolgt ist und alle technischen Voraussetzungen gegeben sind. Falls Voraussetzungen fehlen, ein Vorgehen festlegen, das eine kurze Zeitspanne für sofortige Maßnahmen vorsieht. Bei gravierenden Problemen die Sitzung unmittelbar abbrechen, damit nicht die Zeit von allen für die Behebung der Probleme eines einzelnen aufgewendet wird.

   c. Änderungen an der technischen Infrastruktur, z.B. Software Updates, vorher testen und Neuerungen am Anfang der Sitzung kurz vorstellen.

6. Nach jeder Sitzung bzw. gemeinsamen Terminen sind Erkenntnisse, Vereinbarungen und wichtige Informationen zur Sicherung und Festigung schriftlich, z.B. in Form einer Email, an alle Beteiligten zu verschicken.

Struktur und Ablauf innerhalb der Sitzungen bestimmen

7. Konkrete Regeln sind aufzustellen, die den Ablauf des Entwicklungsprozesses betreffen.

   a. Es wird empfohlen den Entwicklern eine kurze Einführung zu DPP und dessen Vor- und Nachteilen zu geben. Dies würde die Relevanz eines zweiten Entwicklers auch bei unterschiedlichen Erfahrungsstufen (*Expert-Novice*) verdeutlichen und seine Position stärken.

   b. Am Anfang jeder Sitzung die Ziele, die Aufgaben sowie die Strategie für die Bearbeitung, z.B. die Rollenverteilung und Verantwortlichkeiten, nennen. Damit werden Erwartungen geklärt und Missverständnissen wird vorgebeugt.

   c. Rollenwechsel zwischen den Entwicklern regeln, um zu vermeiden, dass der erfahrenere Entwickler zu viel Zeit in der Driver-Rolle verbringt, bzw. dafür sorgen, dass jeder Entwickler regelmäßig die Driver-Rolle übernimmt.

   d. Am Ende die Ergebnisse der Sitzung zusammenfassen, d.h. erreichte und nicht erreichte Ziele sowie erledigte und offene Aufgaben.

   e. Die Planung der nächsten Sitzung, z.B. Aufgaben und Vorbereitung, anhand der Ergebnisse der aktuellen Sitzung, wenn möglich, schon festlegen.

8. Kommunikation und Awareness sind durch geeignete Maßnahmen in den Sitzungen zu erhöhen.
    a. Einen der Entwickler gemäß Punkt 5b) und 7a) die Sitzung eröffnen lassen. Der Betreuer schaltet sich nur ein, wenn die Sitzung ohne seine Hilfe nicht weitergeführt werden kann und wartet dafür möglichst Redepausen, Themenwechsel oder Testläufe ab.
    b. Bei einer Audioverbindung mit mehr als zwei Beteiligten durch Nennung des Namens sicherstellen, dass sich der Empfänger auch angesprochen fühlt.
    c. Für zusätzliche Awareness unabhängig von der Groupware sorgen: (1) indem Punkt 7a) erfüllt wird, (2) indem der Driver sicherstellt, dass der Observer das gleiche sieht wie er selbst und (3) indem der Driver laut kommentiert, was er gerade denkt und tut.

Reflexionen als unterstützende Maßnahme für den Etablierungsprozess gestalten

9. Die Reflexionen sind so zu gestalten, dass sie den Etablierungsprozess so wenig wie möglich durch zusätzlichen Aufwand belasten.
    a. Reflexionen am besten an Sitzungstagen planen, sodass für die Sitzungen und die Reflexionen nur ein Termin gefunden werden muss.
    b. Reflexionen direkt *nach* Sitzungen stattfinden lassen. Die Chance ist größer, dass die Entwickler sich noch an die Geschehnisse der Sitzung erinnern können. Auch werden Verzögerungen des Sitzungsbeginns so vermieden.
    c. Wenn mehrere Sitzungen pro Woche stattfinden, dann empfiehlt es sich, nur eine Reflexion für alle Sitzungen derselben Woche nach der letzten Sitzung durchzuführen. Den Entwicklern spart es Zeit, die sie für andere Projekte brauchen könnten. Der Betreuer hat ausreichend Zeit, Erkenntnisse sinnvoll aufzubereiten und eine intensive Reflexion mit durchdachten Schwerpunkten vorzubereiten, statt viele kleine knappe Reflexionen durchzuführen.
10. Reflexionen, in denen direkte Kommunikation und aktive Beteiligung der Entwickler durch eigene Beiträge und Diskussionen fehlen, können durch folgende Maßnahmen nutzbringend gestaltet werden:
    a. Reflexionen eine gemeinsame visuelle Unterstützung geben, z.B. in Form einer geteilten Datei in Saros mit den wichtigsten Stichpunkten.
    b. Eine offene, leicht zu beantwortende Einstiegsfrage stellen, um die Entwickler zu animieren, zunächst ihnen wichtige Ereignisse einzubringen.
    c. Durch Lob, z.B. über die gute Zusammenarbeit, eine angenehme Atmosphäre schaffen. Die Reflexion ersetzt meistens gemeinsame Zeit, z.B. Kaffee- und Mittagspausen, in denen die Entwickler sich besser kennenlernen können.
    d. Durch Verständnis und Flexibilität, z.B. durch regelmäßiges Zustimmen, eine Atmosphäre schaffen, in der die Entwickler die Möglichkeit haben, ihre Gedanken frei zu äußern.
    e. Ein Leitfaden hilft bei der Moderation, auch ohne Themenvorschläge der Entwickler die Erörterung auf wichtige Aspekte des Entwicklungs- und Etablierungsprozesses zu lenken. Der Leitfaden sollte alle aktuellen Erkenntnisse und wichtigen Daten, z.B. Konflikte aus der Sitzung oder besondere Angaben im Fragebogen, enthalten.
    f. Die Moderation sowie die Vorbereitung der Beiträge für die Reflexion an die Entwickler abgeben. Dabei beachten, dass die Verantwortung für die Reflexion zwischen den Entwicklern gleichmäßig aufgeteilt wird, z.B. indem sie sich bei der Moderation von Reflexionen abwechseln. Dies stärkt vor allem die Position eines Entwicklers, z.B. eines Novizen, der in den Sitzungen sonst weniger zu sagen hat.

5.5 Nutzen von DPP in der industriellen Softwareentwicklung

Dieser Abschnitt erörtert den Nutzen von DPP im industriellen Umfeld. Dafür werden die Vor- und Nachteile von DPP aus den bisherigen Erkenntnissen des Entwicklungs- und Etablierungsprozesses abgeleitet und anschließend gegenübergestellt. Die Erkenntnisse werden mit bereits

bekannten Vor- und Nachteilen von DPP aus anderen Untersuchungen (siehe Kapitel 2) verglichen. Die folgenden Punkte beschreiben die Bedingungen, unter denen DPP eingesetzt wurde, sowie besondere Merkmale der DPP-Sitzungen:

- Es wurden neben DPP keine weiteren agilen Softwareentwicklungsmethoden im Entwicklungsprozess eingesetzt.
- Die Entwickler befanden sich auf unterschiedlichen Erfahrungsstufen. Sie verfügten alle über Erfahrungen als Entwickler, doch nur einer von ihnen, der Experte, hatte bereits Domänenwissen.
- An den Sitzungen waren in der Regel zwei teilweise auch drei Entwickler beteiligt.
- Insgesamt gab es wenig Paarrotation. Der Experte war an allen Sitzungen beteiligt. Die anderen beiden Entwickler wechselten sich ab oder waren beide in den Sitzungen anwesend.
- Es gab nur selten Rollenwechsel zwischen Driver und Observer in Sitzungen.
- Nicht alle Teile des Quellcodes wurden durch DPP erzeugt.
- Es wurde in den Sitzungen für ein späteres Release vorgearbeitet, sodass nur eine vorläufige Spezifikation vorlag und weniger Termindruck herrschte.

Vorteile von DPP im industriellen Umfeld

Die Kommunikation im virtuellen Team basierte vor der Verwendung von DPP hauptsächlich auf Chat- und Emailnachrichten. Sie konnte durch DPP intensiviert werden, da die Entwickler regelmäßig mindestens ein Mal in der Woche gemeinsam eine Sitzung abhielten, in der sie sich über eine Audioverbindung unterhielten und dabei gegenseitig synchron ihre Aktivitäten bei der Entwicklung im Workspace verfolgen konnten.
Weiterhin wurde die Kommunikation durch DPP verstärkt, weil die Entwickler abweichend von ihrem normalen Standardentwicklungsprozess (siehe 4.4.4) (1) gemeinsam Durchsichten durchführten, (2) ihre Aufgaben nicht unabhängig voneinander sondern gemeinsam in den Sitzungen bearbeiteten und (3) sich gegenseitig über Aktivitäten innerhalb und außerhalb der Sitzungen austauschten.
Diese Neuerungen im Entwicklungsprozess erhöhten außerdem auch den Wissenstransfer zwischen den Entwicklern (siehe Abschnitt 5.2.4). Im Gegensatz zum üblichen Entwicklungsprozess wurde existierender Code vorgestellt, oder der Experte erklärte während er weiterentwickelte sein Vorgehen bzw. gab weitere Informationen über den Aufbau und die Funktionalität der Artefakte.
Die verstärkte Kommunikation und auch der erhöhte Wissenstransfer förderten die eingangs als *Pair-Think* und *Pair-Learning* bezeichneten Vorteile von PP bzw. DPP. Durch das Feedback des Experten in den Reviews und durch seine Erklärungen in den Pair-Developing-Sitzungen stellte sich selbstständig ein gemeinsames Verständnis für die Herangehensweise zur Entwicklung ein. Außerdem konnte er während der Erledigung der Aufgaben nebenbei auch noch sein Wissen über die Domäne in die Sitzungen einfließen lassen. Dies wurde in Abschnitt 2.3.1 als einer der Vorteile der Rollenverteilung von *Expert-Novice* bereits eingeführt.
Wie auch in andere Studien so zeigte auch die vorliegende, dass die Entwickler bei DPP besonders Diskussionen über die gemeinsame Herangehensweise und den Entwurf als hilfreich empfinden [WKC00].

Ein weiterer Vorteil von DPP ist die Erhöhung der Qualität der erzeugten Artefakte. Der Observer konnte in Pair-Developing-Sitzungen mehrmals verhindern, dass der Driver Fehler bei der Entwicklung machte (siehe Abschnitt 5.2.4). Dies wird wie schon erläutert in der Literatur als *Pair-Review* bezeichnet, da bereits während der Erzeugung des Quellcodes auf Fehler und Defekte geachtet wird.
Sowohl in Reviews als auch in Pair-Developing-Sitzungen wirkte sich die Rollenverteilung *Expert-Novice* positiv auf die Qualität des Quellcodes aus. Zum einen wurde die Herangehensweise bis ins kleinste Detail vom Experten erklärt und damit genau analysiert, da er kein Domänenwissen bei den anderen Beteiligten voraussetzte. Zum anderen handelte es sich bei den Novizen um

erfahrene Entwickler, die durch gezielte Rückfragen Diskussionen über Entwurfsentscheidungen hervorriefen oder eigene Vorschläge einbrachten.
Die Reviews sorgten für die Erhöhung der Qualität des erzeugten Quellcodes, zum einen weil Missverständnisse über Anforderungen geklärt werden konnten und zum anderen weil dadurch eine zusätzliche außerplanmäßige Prüfung des Quellcodes vor der standardmäßigen Qualitätsprüfung in der Verifikationsphase (siehe Abschnitt 4.4.4) vorgenommen wurde.

Weiterhin merkten die Entwickler an, dass DPP eine Beschleunigung in der verteilten Softwareentwicklung erbrachte, weil dadurch schnelleres Feedback über offene Punkte wie z.B. wichtige Entwurfsentscheidungen und die Klärung von unklaren Anforderungen möglich waren.

Nachteile von DPP im industriellen Umfeld

Einer der Nachteile, der sich zeigte war der erhöhte Ressourcenaufwand. Neben dem doppelten Einsatz von Entwicklern kosteten technische Probleme, mangelnde Vorbereitung und fehlende Routine der Entwickler im Umgang mit der technischen Infrastruktur teilweise bis zu 30 % der Sitzungszeit (siehe Abschnitt 5.2.3) und in seltenen Fällen auch die gesamte Sitzungszeit (siehe Abschnitt 5.2.2).
Bei dem Versuch, einen dritten Entwickler in die Sitzungen zu integrieren zeigte sich, dass sich der Nutzen von DPP nicht einfach auf eine größere Anzahl von Entwicklern erweitern lässt. Der dritte Entwickler hatte Schwierigkeiten, seine Rolle in den Sitzungen zu finden.

Ein weiterer Nachteil war, dass es mehrfach zu Verzögerungen oder Ausfällen von DPP-Sitzungen kam, weil einer der Entwickler nicht daran teilnehmen konnte. Die Gründe dafür waren bspw., dass die Entwickler noch in andere Projekte mit dringlicheren Aufgaben eingebunden waren oder dass technische Probleme bei einem der beteiligten Entwickler auftraten.
Die Terminplanung für die Zusammenarbeit mit DPP wurde durch die verschiedenen Zeitzonen und den Zeitunterschied von mehreren Stunden auf einen kleineren gemeinsamen Zeitraum eingegrenzt (siehe Abschnitt 5.3.4). Teilweise mussten die Entwickler ihren Tagesablauf anpassen, z.B. die Mittagspausen um einige Minuten verschieben. Problematischer war jedoch für die Planung, dass die Entwickler aufgrund ihrer anderen Verpflichtungen nur wenig Zeit zur Verfügung hatten.
Meistens war die Vorgehensweise unstrukturiert, und Ziele, Erwartungen sowie Verantwortlichkeiten der Entwickler bzgl. DPP waren nicht untereinander abgestimmt (siehe Abschnitt 5.2.1). Zum einen hatten die Entwickler wenig oder gar keine Erfahrung mit PP oder DPP. Zum anderen waren vorher keine Regeln für die Zusammenarbeit mit DPP aufgestellt worden.
In den Reviews zeigte sich dies bspw., daran dass die Entwickler anfangs verschiedene Vorstellungen über die Herangehensweise hatten, und dadurch die Zusammenarbeit weniger strategisch und eher stockend war (siehe Abschnitt 5.2.4).
In Pair-Developing-Sitzungen machte sich dies bemerkbar, weil der Novize Schwierigkeiten hatte, sich aktiv in die Sitzungen einzubringen. Anfangs fühlte er sich durch den Wissensunterschied und die Anwesenheit des zweiten Entwicklers bei der Entwicklung verunsichert. Später hatte er Schwierigkeiten, sich eine sinnvolle Rolle in den Sitzungen einzuräumen. Sowohl er als auch der Experte waren mit den seltenen Rollenwechseln unzufrieden, taten sich aber schwer, dafür eine passende Lösung zu finden. An diesem Punkt stellt sich die Frage, in welchem Maß die Entwickler Interesse daran hatten, überhaupt eine Lösung zu finden. Zu klären bliebe ferner inwiefern es eine Rolle für DPP im industriellen Umfeld spielt, ob sich die Entwickler freiwillig zu der Zusammenarbeit entschließen oder dafür ausgewählt werden. Aus zeitlichen Gründen konnte dieser Aspekt im Rahmen dieser Arbeit nicht weiter erörtert werden.

Die räumliche Distanz schaffte auf verschiedenen Ebenen der Zusammenarbeit Probleme. Wie vorher schon erwähnt fehlte den Entwicklern teilweise die Abstimmung von Zielen, Erwartungen und Verantwortlichkeiten, was ihnen laut einer Studie über DPP wahrscheinlich leichter gefallen wäre, wenn sie auch gemeinsame Bürozeiten und Pausen gehabt hätten [Yap05].

Auch fehlende Awareness bei der Zusammenarbeit führte zu Problemen (siehe Abschnitt 5.2.4). Zum einen fehlte in Pair-Developing-Sitzungen manchmal Feedback darüber, ob der Observer den Aktivitäten des Drivers überhaupt folgen konnte, z.B. wenn der Observer nach der Position des Drivers fragen musste, oder er durch äußere Einflüsse abgelenkt wirkte. Wie in anderen Studien bereits beobachtet wurde, empfanden die Entwickler die Notwendigkeit, ununterbrochen miteinander sprechen zu müssen und alle ihre Aktivitäten kommentieren zu müssen, als belastend. Eine zusätzliche Videoverbindung hätte in diesem Fall evtl. die Zusammenarbeit durch nonverbales Feedback erleichtern können.

Insgesamt war auffällig, dass die Entwickler Schwierigkeiten hatten, sich auf die neue Praktik einzustellen. Die als *Pair-Jelling* bezeichnete Eingewöhnungsphase dauerte wesentlich länger als erwartet. Studien über PP ergaben, dass die betreffenden Entwickler nicht mehr als wenige Tage brauchten, um sich auf die neue Praktik einzustellen [WK00].
Entgegen meinen Erwartungen gaben die Entwickler an, dass sie sich in der Zusammenarbeit nur wenig etabliert fühlten. Sie taten sich außerdem schwer damit, offen miteinander zu sprechen oder sich gegenseitig zu kritisieren.
Über die Gründe dafür kann an dieser Stelle nur spekuliert werden. Beispielsweise waren die Entwickler teilweise neu im Unternehmen und mussten sich nicht nur an eine neue Arbeitsumgebung, sondern gleichzeitig auch auf eine neue Praktik einstellen, über die sie im Vorfeld keine Kenntnis hatten. Es könnten ebenfalls Schwierigkeiten in der Teambildung als Ursache in Betracht gezogen werden. In anderen Studien wurden zudem kulturelle Unterschiede, Sprachschwierigkeiten oder Probleme mit sozialer Interaktion auf die Distanz als Hindernisse identifiziert [Yap05, BM02, OO00]. Doch gibt es auch Studien, die belegen, dass nicht jeder Entwickler die Zusammenarbeit über eine räumliche Distanz oder in Form von PP bzw. DPP für sinnvoll hält [SWN+03, Yap05, WKC00]. Sie führen bspw. als Grund an, dass manche Entwickler lieber alleine arbeiten würden. Die genaue Ergründung der Problematik von *Pair-Jelling* ist im Rahmen künftiger Untersuchungen eingehender zu untersuchen.

Zusammenfassung

Durch die Anwendung von DPP konnten die Kommunikation und der Wissenstransfer im virtuellen Team sowie die Qualität des Quellcodes erhöht werden. Zusätzlich stellte es sich als eine Methode heraus, die das Feedback des verteilten Teams beschleunigen kann.
Es zeigte sich zwar ein Nachteil in Form des hohen Ressourcenaufwandes, jedoch konnte dieser teilweise schon während des Projektes gesenkt werden. Der Einsatz von zwei Entwicklern hätte sich auch bei einer anderen Form der Zusammenarbeit kaum vermeiden lassen, da die Aufgabe darin bestand, die neu angestellten Entwickler in das Projekt einzuarbeiten. Im Gegenteil konnte durch DPP die Erklärung gleichzeitig mit der Erzeugung von neuen Funktionalitäten einhergehen. Mehrfach kam die Zusammenarbeit ins Stocken, da einer der Entwickler dringendere Aufgaben aus anderen Projekten zu erledigen hatte. Nur am Rande hatte dies mit den unterschiedlichen Zeitzonen zu tun. Es ist möglich, dass die Wahl von weniger dringlichen Aufgaben als Inhalte für das DPP-Projekt ein Grund für die fehlende Kontinuität war. Es zeigten sich außerdem verschiedene Nachteile wie z.B. fehlende Struktur und Strategie sowie Schwierigkeiten im Umgang mit der räumlichen Distanz, die sich auf die Unerfahrenheit der Entwickler mit DPP zurückführen lassen. Es ist gut möglich, dass sich diese sowie weitere Schwierigkeiten der Entwickler, die sich aus der fehlenden Routine in der Zusammenarbeit im neugebildeten Team ergaben, mit dem Ende der Eingewöhnungsphase von selbst verringert hätten.
Für ein virtuelles Team, das gemeinsam an Projekten arbeitet, die Wissen über den Quellcode der anderen Teammitglieder voraussetzen, überwiegen jedoch Vorteile einer Anwendung von DPP. Die meisten Nachteile können langfristig verringert oder ausgeräumt werden.

6 Resümee

Wie bereits in der Einleitung erläutert gab es zwei Fragen, anhand derer die Etablierung von DPP im industriellen Umfeld untersucht werden sollte:

1. Welche Probleme können bei der Etablierung von DPP im industriellen Einsatz auftreten, und wie lassen sich diese durch Anpassung des Etablierungsprozesses beheben bzw. verringern?
2. Welchen Nutzen hat DPP im Rahmen industrieller Softwareentwicklung?

Nach einer ausführlichen Recherche über PP und DPP wurden verschiedene Forschungsmethoden und Maßnahmen eingesetzt, um den Entwicklungs- und den Etablierungsprozess zu optimieren und die zu untersuchenden Fragen zu beantworten. Primär diente zur Etablierung von DPP eine strukturierte Form der Handlungsforschung (CAR), wodurch der Forschungsprozess in Zyklen unterteilt, schrittweise Erkenntnisse gesammelt und weitere Maßnahmen geplant werden konnten. Des Weiteren wurde eine qualitative Inhaltsanalyse angewendet, um detaillierte Erkenntnisse über den Entwicklungs- und Etablierungsprozess bzw. über den Nutzen von DPP im industriellen Umfeld zu erörtern.

Die Untersuchung der ersten Forschungsfrage ergab, dass sich einige Probleme bei der Etablierung von DPP einstellten, die nur teilweise im Verlauf des Projektes behoben werden konnten. Ein erhebliches Problem stellte der anfänglich sehr hohe Ressourcenaufwand dar. Dieser konnte jedoch durch verschiedene Maßnahmen erheblich reduziert werden. Weiterhin zeigten sich Schwierigkeiten bei der Terminplanung und Verzögerungen bzw. Ausfälle von Sitzungen, weil die Entwickler öfter dringlichere Aufgaben aus anderen Projekten zu erledigen hatten. Die vorerst unstrukturierte Vorgehensweise bei der Anwendung von DPP konnte während des Etablierungsprozesses teilweise durch Anpassungen strukturierter gestaltet werden. Jedoch konnten nicht alle Konflikte, z.B. aufgrund fehlender Awareness oder hinsichtlich der Verteilung der Verantwortlichkeiten, gelöst werden.
Die eingesetzten Forschungsmethoden und Maßnahmen, darunter vor allem Reflexionen, konnten dazu beitragen, relevante Informationen zu erheben und den Etablierungsprozess zu optimieren. Sie wurden von den Entwicklern zwar als zusätzlicher Aufwand empfunden und wenig unterstützt aber von diesen dennoch angenommen.
Aus dem Etablierungsprozess ließ sich weiterhin die Erkenntnis ziehen, dass entgegen den Vermutungen der Forscher vor zehn Jahren es heute verschiedene Technologien gibt, die DPP ausreichend unterstützen. Saros hat sich in dieser Hinsicht als Groupware bewährt.

Bezüglich der zweiten Forschungsfrage ergab die qualitative empirische Untersuchung des Entwicklungs- und Etablierungsprozesses, dass trotz Schwierigkeiten mit der räumlichen Distanz und trotz einer fehlenden Routine in der Zusammenarbeit die Ziele der Entwickler im Etablierungsprozess durch DPP erreicht werden konnten. Weiterhin zeigte sich, dass der Entwicklungsprozess im virtuellen Team durch intensivere Kommunikation und erhöhten Wissenstransfer verbessert werden konnte. Die Entwickler empfanden es ebenfalls als angenehm, dass sich durch DPP das Feedback über offene Punkte beschleunigte. Außerdem konnte die Qualität der erzeugten Software sowohl durch *Pair-Review* als auch durch die Einführung von zusätzlichen Reviews im Entwicklungsprozess erhöht werden.

7 Ausblick

In dieser Arbeit wurden viele Erkenntnisse über den konkreten Etablierungsprozess und DPP im Allgemeinen aufgezeigt. Jedoch sind einige weitere Aspekte in anderem Rahmen zu untersuchen.

Es wurden viel mehr Daten durch Aufzeichnungen von Sitzungen, Reflexionen und Erhebungen in Fragebogen und Einzelgesprächen erhoben, als in dieser Arbeit verwendet werden konnten. Einige interessante Phänomene konnten nicht in die vertiefende Analyse einbezogen werden. Dazu gehören z.B. Erörterungen darüber, wie und wodurch bei DPP Zeit gespart werden kann, oder wie der Wissenstransfer im Detail vor sich ging.

Außerdem ist unklar, inwiefern die für die Sitzungen gewählten Aufgaben die Ergebnisse beeinflussten und ob andere ‘dringlichere‘ Aufgaben als Sitzungsinhalte den Ablauf und die Ergebnisse verändert hätten. Ein weiteres Projekt könnte in dieser Hinsicht weitere Erkenntnisse erbringen.
Im Zuge dessen könnte zusätzlich zum praxisnahen Inhalt der Etablierungsprozess durch eine erweiterte technische Infrastruktur, z.B. mit Videoübertragung, und der Anleitung aus Abschnitt 5.4 optimiert werden.

Es könnte von Interesse sein, in zukünftigen Untersuchungen die parallele Mehrbenutzer-Unterstützung von Saros verstärkt einzusetzen oder die Beteiligung von mehr als zwei Entwicklern zu erforschen. Dadurch könnten sich weitere Vorteile von verteilter kollaborativer Softwareentwicklung gegenüber PP zeigen, z.B. bei unabhängiger aber synchroner und paralleler Zusammenarbeit bzw. für Schulung von mehr als einem Entwickler.

Überdies gibt es bisher nur sehr wenige Studien, die sich mit interdisziplinären Betrachtungsweisen von DPP beschäftigen. Es wäre in diesem Zusammenhang für virtuelle Teams von global agierenden Unternehmen vermutlich auch interessant zu erfahren, wie sich wirtschaftliche, kulturelle und soziale Einflüsse auf DPP auswirken.

8 Literaturverzeichnis

[Ave04] Aveling, B. (2004). XP Lite Considered Harmful? *Proceedings. 5th International Conference XP* , pp. 94-103.

[BWG02] Baheti, P., Williams, L., Gehringer, E., & Stotts, D. (2002). Exploring Pair Programming in Distributed Object-Oriented Team Projects. *OOPSLA - Object-Oriented Programming Systems, Languages, and Applications* .

[BW84] Basili, R., & Weiss, D. M. (1984). A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering, vol. SE-10, no.6* , pp. 728-738.

[BCR94] Basili, V., Caldiera, G., & Rombach, H. D. (1994). The Goal Question Metric Approach. In *Encyclopedia of Software Engineering* (S. 528-532). John Wiley & Sons.

[Bec99] Beck, K. (1999). Embracing Change with Extreme Programming. *Computer, vol. 32, no. 10*, (pp. 70-77).

[Boe76] Boehm, B. (1976). Software Engineering. *IEEE Transactions on Computer, Vol. 25* , pp. 1226-1241.

[BM02] Bradner, E., & Mark, G. (2002). Why Distance Matters: Effects on Cooperation, Persuasion and Deception. *Proceedings of CSCW* .

[BJ05] Braithwaite, K., & Joyce, T. (2005). XP Expanded: Distributed Extreme Programming. *6th International Conference on eXtreme Programming and Agile Processes in Software Engineering*, (pp. 180-188).

[Brü00] Brüsemeister, T. (2000). *Qualitative Forschung: Ein Überblick.* Wiesbaden: Westdeutscher Verlag.

[Camtasia] *Camtasia*. (2009). Abgerufen am 27. 09 2009 von http://www.techsmith.de/camtasia.asp

[CCV03] Canfora, G., Cimitile, A., & Visaggio, C. (2003). Lessons learned about distributed pair programming: what are the knowledge needs to address ? *In Proc 12th IEEE Int. Wsh. on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE03)* , S. 314-319.

[Coc04] Cockburn, A. (2004). Crystal Clear. Addison-Wesley.

[CW01] Cockburn, A., & Williams, L. (2001). The costs and benefits of pair programming. *Extreme Programming Examined, Addison Wesley* , pp. 223-248.

[DL04] Damina, D., & Lanubile, F. (2004). The 3rd International Workshop on Global Software Development. *Proceedings of the 26th International Conference on Software Engineering (ICSE'04).*

[DMK04] Davison, R., Martinsons, M., & Kock, N. (Januar 2004). Principles of canonical action research. *Information Systems Journal, Vol. 14, No. 1.* , S. 65-86.

[DD06] Denning, P., & Durham, R. (2006). Innovation as language action. *COMMUNICATIONS OF THE ACM, vol. 49, 5* , S. 47-52.

[Dje06] Djemili, R. (2006). Entwicklung einer Eclipse-Erweiterung zur Realisierung und Protokollierung verteilter Paarprogrammierung. *Diplomarbeit* . Freie Universität Berlin.

[DOS07] Djemili, R., Oezbek, C., & Salinger, S. (2007). Saros: Eine Eclipse-Erweiterung zur verteilten Paarprogrammierung. *Software Engineering (Workshops), Vol. 106 of LNI* , S. 317-320.

[DB92] Dourish, P., & Bellotti, V. (1992). Awareness and coordination in shared workspaces. *Proceedings of the ACM Conference on CSCW* , S. 107-114.

[Dwo09] Dworschak, M. (06. 07 2009). Windows aus der Asche. *Der Spiegel, Ausgabe 28* , S. 114-116.

[Eclipse] *Eclipse*. (2009). Abgerufen am 09. 23 2009 von http://www.eclipse.org/projects/

[EGR91] Ellis, C., Gibbs, S., & Rein, G. (1991). Groupware: some issues and experiences. *Communications of the ACM Vol.34 (1)* , S. 39-58.

[GM02] George, B., & Mansour, Y. M. (2002). A Multidisciplinary Virual Team. *Proceedings Systemics, Cybernatics and Informatics (SCI).*

[GoogleDocs] *Google Docs.* (kein Datum). Abgerufen am 23. 09 2009 von http://www.google.com/google-d-s/intl/de/tour2.html

[GG99] Gutwin, C., & Greenberg, S. (1999). The effects of workspace awareness support on the usability of real-time distributed groupware. *ACM Transactions on Computer-Human Interaction 6 (3)* , S. 243-281.

[GGR96] Gutwin, C., Greenberg, S., & Roseman, M. (1996). Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation. *People and Computers XI* , S. 281-298.

[Han04] Hanks, B. (2004). Tool support for distributed pair programming: An empirical study. *Proceedings of XP/Agile Universe 2004: 4th Conference on Extreme Programming and Agile Methods* .

[Hol75] Holm, K. (1975). *Die Befragung 1:Der Fragebogen – Die Stichprobe.* München: Francke-Verlag.

[Hus87] Huschke-Rhein, R. (1987). *Qualitative Forschungsmethoden und Handlungsforschung II.* Köln: Rhein-Verlag.

[Jac09] Jacob, C. (2009). *Weiterentwicklung eines Werkzeuges zur verteilten, kollaborativen Softwareentwicklung.*

[MM02] Maurer, F., & Martel, S. (2002). *Process Support for Distributed Extreme Programming Teams.* University of Calgary, Canada.

[Mos77] Moser, H. (1977). *Praxis der Aktionsforschung. Ein Arbeitsbuch.* München: Kösel-Verlag.

[MPT05] Müller, M. M., Padberg, F., & Tichy, W. F. (2005). Ist XP etwas für mich? Empirische Studien zur Einschätzung von XP. *Tagungsband Software Engineering 64* , S. 217-228.

[Mumble] *Mumble*. (2009). Abgerufen am 27. 09 2009 von http://mumble.sourceforge.net/

[Netmeeting] *Microsoft Netmeeting.* (2009). Abgerufen am 23. 09 2009 von http://microsoft-netmeeting.softonic.de/

[Netviewer] *Netviewer Meet*. (2009). Abgerufen am 27. 09 2009 von http://www.netviewer.de/produkte/netviewer-meet/

[Nos98] Nosek, J. T. (1998). The Case for Collaborative Programming. *Communications of the ACM 41 Nr.3*, (S. 105-108).

[OO00] Olson, G. M., & Olson, J. S. (2000). Distance Matters. *Proceedings Human Computer, Vol. 25*, (S. 139-179).

[OpenVPN] *OpenVPN*. (2009). Abgerufen am 27. 09 20096 von http://www.openvpn.net/

[OSI] *OSI*. (2009). Abgerufen am 23. 09 2009 von http://www.opensource.org/

[PEP] *PEP.* (2009). Abgerufen am 23. 09 2009 von - Pair Eclipse Programming: http://pep-pp.sourceforge.net/

[RZ04] Reeves, M., & Zhu, J. (2004). Moomba - A Collaborative Environment for Supporting Distributed Extreme Programming in Global Software Development. *Lecture Notes in Computer Science', Vol. 3092* , S. 38-50.

[Ric05] Richards, L. (2005). *Handling Qualitative Data - A practical Guide.* London : Sage.

[Rod96] Rodden, T. (1996). Population the Application: A Model of Awareness for Cooperative Applications. *Proceedings of International Conference on Computer Supported Cooperative Work* , S. 87-96.

[SP08] Salinger, S., & Prechelt, L. (2008). What happens during Pair Programming? *Proceedings of the 20th Annual Workshop of Psychology of Programming Interest Group.*

[Sangam] *Sangam.* (2009). Abgerufen am 23. 09 2009 von http://sangam.sourceforge.net/

[Saros] *Saros*. (2009). Abgerufen am 23. 09 2009 von https://www.inf.fu-berlin.de/w/SE/DPP

[Sch93] Scholl, A. (1993). *Die Befragung als Kommunikationssituation.* Opladen: Westdeutscher Verlag.

[SL08] Schümmer, T., & Lukosch, S. (2008). Supporting the Social Practices of Distributed Pair Programming. *Groupware: Design, Implementation, and Use: 14th International Workshop, CRIWG 2008* , S. 83-98.

[SS01] Schümmer, T., & Schümmer, J. (2001). Support for Distributed Teams in eXtreme Programming. *eXtreme Programming Examined* , S. 355-378.

[SR03] Stephens, M., & Rosenberg, D. (2003). *Extreme Programming Refactored: The Case Against XP.* Apress.

[SWN03] Stotts, D., Williams, L., Nagappan, N., Baheti, P., Jen, D., & Jackson, A. (2003). Virtual Teaming: Experiments and Experiences with Distributed Pair Programming. *XP/Agile Universe* , S. 129-141.

[SC90] Strauss, A., & Corbin, J. (1990). *Basics of Qualitative Research - Grounded Theory Procedures and Techniques.* London: Sage.

[SubEthaEdit] *SubEthaEdit*. (2009). Abgerufen am 23. 09 2009 von http://www.codingmonkeys.de/subethaedit/

[Teles] *Teles*. (2009). Abgerufen am 23. 09 2009 von http://www.teles.de/de/about-us/at-a-glance/

[Tel08] TelesAG. (2008). *Konzernabschluss und Konzernlagebericht.*

[Tuc65] Tuckman, B. (1965). Developmental sequences in small groups. *Psychological Bulletin, 63* , S. 348-399.

[UltraVNC] *UltraVNC*. (2009). Abgerufen am 23. 09 2009 von http://www.uvnc.com/index.html

[VAC] *Virtual Audio Cable*. (2009). Abgerufen am 27. 09 2009 von http://software.muzychenko.net/eng/vac.html#features

[VirtualBox] *Virtual Box*. (2009). Abgerufen am 27. 09 2009 von http://www.virtualbox.org/wiki/Virtualization

[Wie01] Wiegers, K. (2001). When Two Eyes Aren't Enough. *Software Development, vol. 9, no. 10* .

[Wil00] Williams, L. (2000). The Collaboration Software Process. *Dissertation* .

[WK00] Williams, L., & Kessler, R. (2000). All I Ever Needed to Know about Pair Programming. *Communications of the ACM 43 Nr.5* , S. 108-114.

[WK02] Williams, L., & Kessler, R. (2002). *Pair Programming Illuminated.* Addison Wesley.

[WKC00] Williams, L., Kessler, R., & Cunningham, W. (2000). Strengthening the Case for Pair Programming. *IEEE Software 17 Nr. 4* , S. 19-25.

[XecliP] *XecliP.* (2009). Abgerufen am 23. 09 2009 von http://sourceforge.net/projects/xeclip/

[XPairtise] *XPairtise.* (2009). Abgerufen am 23. 09 2009 von - Pair Programming for Eclipse: http://sourceforge.net/projects/xpairtise/

[Yap05] Yap, M. (2005). Follow the sun: distributed extreme programming development.

9 Glossar

AG SE	-	Arbeitsgruppe Software Engineering
CAR	-	Kanonische Handlungsforschung/Canonical Action Research
CDR	-	Call Data Records
CET	-	mitteleuropäische Zeit/Central European Time
CSCW	-	Computer Supported Cooperative Work
DPP	-	verteilte Paarprogrammierung/Distributed Pair Programming
Driver	-	Rolle des Entwicklers, der bei PP/DPP die Tastatur und die Maus bedient
Driverwechsel	-	Übergabe der Rolle des Drivers in PP/DPP-Sitzungen
F&E	-	Forschung und Entwicklung
FAD	-	Funktionales Anforderungsdokument/Functional Ápplication Description
GQ	-	Allgemeiner Fragebogen/General Questions
GQM	-	Forschungsmethode für Befragungen/Goal-Question-Metric
Head of Development	-	Leiter der Abteilung F&E
MRS	-	Fragebogen nach der Sitzung/Questionnaire Most Recent Session
Observer	-	Rolle des Entwicklers, der bei PP/DPP die Aktivitäten des Drivers beobachtet
PLM	-	Product Line Manager, Verantwortlicher für das Produkt
PP	-	Paarprogrammierung/Pair Programming
SIT	-	Standard Indian Time
SNMP	-	Netzwerkprotokoll/Simple Network Management Protocol
STD	-	Testdokument/ Standard Test Discription
TAD	-	Technisches Anforderungsdokument/Technical Application Description
US	-	Fragebogen vor der Sitzung/Questionnaire Upcoming Session
Verfolgermodus	-	Saros-Funktionalität zur Verfolgung der Aktivitäten des Drivers/Follow Mode

Anhang

A. GQM-Planung

Was ist das Ziel?

1. Zu bestimmen, wie verteilte Paarprogrammierung nützlich/erfolgreich eingesetzt werden kann
2. Zu identifizieren, welche Probleme auftreten können
3. Herauszufinden, wie man mit Problemen umgehen kann
   - GQM Template:

Analyze: zu messendes Objekt	verteilte PP Sitzungen bei verschiedenen Sitzungstypen: Reviews Anlernen/Wissenstransfer Verteilte PP/gemeinsam entwickeln Andere Daten, die nichts mit der Sitzung zu tun haben
For the purpose of: Charakterisieren, Verstehen, Bewerten, Vorhersagen, Verbessern	Verstehen der Abläufe der verteilten Paarprogrammierung Einschätzen der Situation: Wer hat was wie verstanden? Welche Wichtigkeit haben die Situationen? Verbessern der Sitzungssituation
With respect to: Fokus	Nützlichkeit/Erfolg verteilter PP bewerten Was macht es nützlich? Schaffen die Entwickler das? Probleme (Attribute) in der Sitzung entdecken
From the viewpoint of: Beteiligte	Teles Mitarbeiter
In the context of: Kontext, Umgebung der Messung	Teles IT-Unternehmen, Abteilung Forschung und Entwicklung zwischen Wien und Indien

GQM-Fragen zum Ziel: (Was möchte ich wissen, um ans Ziel zu kommen)

1. Zu Nützlichkeit/Erfolg:
   Was ist eine erfolgreiche Sitzung?
   Welche Erwartungen/Befürchtungen bestehen bzgl. dieser Sitzung?
   Wie wird Erfolg von den Beteiligten eingestuft? Was ist besser?
   Wodurch zeichnet sich eine erfolgreiche Sitzung aus?
   Welche Aspekte bringen die Nützlichkeit/den Erfolg verteilter PP Sitzungen hervor? (Aspekte sind Erfolgskomponenten)
   Haben alle dasselbe Verständnis von einer erfolgreichen Sitzung?
   Wurden die Ziele/gewünschten Ergebnisse erreicht?

2. Zu Problemen:
   Welche Erwartungen wurden nicht erfüllt bzgl. verteilter PP?
   Welche Befürchtungen bestehen bzgl. verteilter PP?
   Haben Sie Abneigung gegenüber verteilter PP?
   Was halten Sie von verteilter PP?
   Gibt es technische Probleme mit Saros oder Benutzerfreundlichkeitsprobleme, die die Qualität der Sitzung beeinflussen?

3. Zu Umgang mit Problemen:
   Haben die Entwickler Vorstellungen, wie man mit den Problemen umgehen kann?

GQM: Was sind die Maße? Wie kann man das messen?

- Alles, was die Entwickler als Erfolg/nützlich identifizieren und was allgemein erfolgreich/nützlich ist, z.B. Aufgabe beenden, Ziel erreichen, Defekte erkennen und eliminieren.
- Alles, was die Entwickler als Problem identifizieren und was allgemein als Problem angesehen werden kann, z.B. Missverständnisse, technische Ausfälle und Störungen von außen.
- Das Maß ist der Fragebogen.

Basisfragebogen (General Questions): Allgemeine Voraussetzungen

- **Basisfragen** zur Situation (Entwickler, Erfahrung, persönliche Einstellung zu DPP, d.h. Erwartungen und Befürchtungen)

Fragebogen 1 (Session Plan): Was ist geplant, was wird erwartet, was wird befürchtet?

- Fragen über die **geplante Aufgabe** und ihre Verbindung zu **DPP**
- Erwartungen vor der Sitzung und gewünschte Ergebnisse nach der Sitzung abfragen/Fragen nach dem **Erfolgsmaß**
- Wie wird Misserfolg von den Beteiligten beurteilt?

Fragebogen 2 (Session Review) Wurde das Ziel erreicht?

- Abfragen, ob Befürchtungen und Erwartungen (aus der persönlichen Einstellung) an DPP sich bewahrheitet haben.
- Fragen über die **erledigte Aufgabe** und ihre Verbindung zu **DPP**
- Erfüllte oder nicht erfüllte Erwartungen nach der Sitzung und Ergebnisse nach der Sitzung abfragen/Fragen nach dem **Erfolgsmaß**

B. Fragebogen

General Questions

Developer's name: Date:

1. Since when have you worked as a professional software developer?

   year(s) month(s)

2. Since when have you worked for your company?

Company name:

   year(s) month(s)

3. What is your preferred proceeding (initial steps) when you deal with new topics, e.g. ask someone who has experience with it, do research (books, journals, etc.), ask an expert?

4. Have you ever done collaborative software development*?

☐ yes ☐ no

If yes, since when have you worked regularly (several times a month) in collaborative software development* sessions?

   year(s) month(s)

5. What do you think about collaborative software development* in general?

6. What constitutes a successful collaborative software development* session from your point of view?

7. What constitutes an unsuccessful collaborative software development* session in your opinion?

Note:

* By collaborative software development we mean any development (coding, testing, reviewing, etc.), where two or more developers work jointly, concurrently and simultaneously on the same code base or artifact(s).

Questions about the upcoming session

Company:	Teles AG	Date:
Name:		Co-Developers:
Start time:	10:30	Planned Duration:

Please note: the following questionnaire refers to the upcoming programming session, not to collaborative software development* in general.

1. Of what types are the upcoming tasks, please classify:

☐ Starting new task	☐ Continue task	☐ Debugging
☐ Code review	☐ Hand over	☐ Refactoring
☐ Developing test cases	☐ Testing	☐ Other:

2. What will be the general procedure of the session?

3. Please give a short description of the upcoming task(s) and estimate your ability to accomplish it (skill level):

Estimated Duration (hh:mm) h=hrs,m=min:

Skill levels: 1=expert, 2=advanced knowledge, 3=basic knowledge, 4= little knowledge, 5=newbie

1. Task: Skill level: Duration:
2. Task: Skill level: Duration:
3. Task: Skill level: Duration:

4. Please specify the challenges of the task(s):

5. How have you prepared for the session?

6. How well established do you feel with your co-developer(s):
   1=very well, 2=well, 3 =fair, 4=never worked together?

   1. Name of the co-developer: estimate:

   Why?

2. Name of the co-developer: estimate:

Why?

7. Please explain why the tasks are suitable for collaborative software development*?

8. What difficulties do you expect using collaborative software development* in this session?

9. What benefit do you expect from this collaborative software development* session?

10. Assign priorities 1 to 6 to the following expressions according to their importance/relevance for a successful session:
    Note: each priority can only be applied once.
    1=very important, 2= important, 3=somewhat important,
    4=of little importance, 5=not as important, 6=least important

    - to finish task(s) in time.

    - to achieve the goal(s).

    - to receive all important information

    - to transfer all important information

    - no technical problem(s) occur or disturb the session

    - to detect and/or eliminate defect(s)

Note:

* By collaborative software development we mean any development (coding, testing, reviewing, etc.), where two or more developers work jointly, concurrently and simultaneously on the same code base or artifact(s).

Questions about the upcoming session

Company:	Teles AG	Date:
Name:		Co-Developers:
Start time:	10:30	Planned Duration:

Please note: the following questionnaire refers to the upcoming programming session, not to collaborative software development (CSD)* in general.

1. Of what types are the upcoming tasks, please classify:

☐ Starting new task	☐ Continue task	☐ Debugging
☐ Code review	☐ Hand over	☐ Refactoring
☐ Developing test cases	☐ Testing	☐ Other:

2. Please give a short description of the upcoming task(s) and estimate your ability to accomplish it (skill level):

Estimated Duration (hh:mm) h=hrs,m=min:

Skill levels: 1=expert, 2=advanced knowledge, 3=basic knowledge, 4= little knowledge, 5=newbie

4. Task: Skill level: Duration:

5. Task: Skill level: Duration:

6. Task: Skill level: Duration:

3. How well established do you feel with your co-developer(s):
1=very well, 2=well, 3 =fair, 4=never worked together?

1. Name of the co-developer: estimate:

2. Name of the co-developer: estimate:

4. Please specify the challenges of the task(s) and how you have prepared for the session?

5. What benefit do you expect from this CSD session?

6. What difficulties do you expect using CSD in this session?

7. Assign priorities 1 to 6 to the following expressions according to their importance/relevance for a successful session: Note: each priority can only be applied once.

1 = very important	**2** = important	**3** = somewhat important
4 = of little importance	**5** = not as important	**6** = least important

- to finish task(s) in time.

- to achieve the goal(s).

- to receive all important information

- to transfer all important information

- no technical problem(s) occur or disturb the session

- to detect and/or eliminate defect(s)

Note:

* By collaborative software development (CSD) we mean any development (coding, testing, reviewing, etc.), where two or more developers work jointly, concurrently and simultaneously on the same code base or artifact(s).

Questions about the most recent session

Company: Teles AG Date:
Developer's name: Co-Developer(s):

Please note: the following questionnaire refers to the most recent programming session, not to collaborative software development in general.

For additional information not covered by these questions, please use comment fields 11 or 12.

1. Did the session produce the benefits you expected from collaborative software development?

☐ yes ☐ no

Why?

2. Did any of the difficulties you expected due to collaborative software development arise during the session?

☐ yes ☐ no

Why?

3. Did new difficulties arise during the session?

☐ yes ☐ no

If yes, please explain:

4. Please decide to which degree collaborative software development helped you to accomplish the session task:

☐ very much ☐ a good deal ☐ somewhat ☐ very little

5. Did you accomplish the task faster than expected?

☐ faster ☐ as expected ☐ slower than expected

Why?

6. Please break up the session into chronological phases and indicate how much time you needed for these phases (you may indicate the duration in percent or minutes):

Phase 1: Duration:

Phase 2: Duration:

Phase 3: Duration:

Phase 4: Duration:

Phase 5: Duration:

7. Please decide which of these phases were the most important for achieving the planned session goal(s) and give reasons for your decision:

i. Phase:

Reason:

ii. Phase:

Reason:

8. Were there phases or parts of tasks when you wished to carry on alone?

☐ yes ☐ no

If yes, please indicate which task it was and why:

i. Task :

Reason:

ii. Task :

Reason:

iii. Task :

Reason:

9. Were there phases in which you found collaborative software development particularly helpful? Please indicate which phases these were and give reasons if possible:

i. Phase:

Reason:

ii. Phase:

Reason:

iii. Phase:

Reason:

10. Do you think the session was accomplished successfully?

☐ yes ☐ no

Please indicate in which manner it was successful:

☐ It was no total failure.
☐ We finished task(s) in time.
☐ We achieved the goal(s).
☐ We solved important problem(s).
☐ We worked productively.
☐ other:
☐ I received all important information.
☐ I transferred all important information.
☐ We detected and/or eliminated defect(s).
☐ No technical problem(s) occurred/disturbed.
☐ We worked better than expected.

Please indicate in which manner it was <u>not</u> successful:

☐ It was a total failure.
☐ We did not finish task(s) in time.
☐ We did not achieve the goal(s).
☐ We did not solve important problem(s).
☐ We did not work productively.
☐ other:
☐ I did not transfer all important information.
☐ I did not transfer all important information.
☐ We detected a lot of/could not eliminate defect(s).
☐ Comprehension problems/misunderstandings occurred.
☐ Technical problem(s) disturbed the session.

11. Do you have any comments or requirements concerning the usage of Saros?

12. Do you have any other comments, apprehensions or expectations about planned or accomplished sessions?

<u>Note:</u>

* By collaborative software development we mean any development (coding, testing, reviewing, etc.), where two or more developers work jointly, concurrently and simultaneously on the same code base or artifact(s).

Questions about the most recent session

Company: Teles AG Date:
Developer's name: Co-Developer(s):

Please note: the following questionnaire refers to the most recent programming session, not to collaborative software development in general.
For additional information not covered by these questions, please use comment field 9.

1. Did the session produce the benefits you expected from collaborative software development?
☐ yes ☐ no
Why?

2. Did new or known difficulties due to collaborative software development arise during the session?
☐ yes ☐ no
Why?

3. Please decide to which degree collaborative software development helped you to accomplish the session task:
☐ very much ☐ a good deal ☐ somewhat ☐ very little

4. Did you accomplish the task faster than expected?
☐ faster ☐ as expected ☐ slower than expected
Why?

5. Please decide which phases were most important for achieving the planned session goal(s) and give reasons for your decision:
i. Phase: Duration:
Reason:
ii. Phase: Duration:
Reason:
iii. Phase: Duration:
Reason:

6. Were there phases or parts of tasks when you wished to carry on alone?
☐ yes ☐ no
If yes, please indicate which task it was and why:
i. Task :
Reason:
ii. Task :
Reason:

7. Were there phases in which you found collaborative software development particularly helpful? Please indicate which phases these were and give reasons if possible:

i. Phase:
Reason:
ii. Phase:
Reason:
iii. Phase:
Reason:

8. Do you think the session was accomplished successfully?
☐ yes ☐ no

Please indicate in which manner it was successful:

☐ It was no total failure.	☐ I received all important information.
☐ We finished task(s) in time.	☐ I transferred all important information.
☐ We achieved the goal(s).	☐ We detected and/or eliminated defect(s).
☐ We solved important problem(s).	☐ No technical problem(s) occurred/disturbed.
☐ We worked productively.	☐ We worked better than expected.
☐ other:	

Please indicate in which manner it was <u>not</u> successful:

☐ It was a total failure.	☐ I did not transfer all important information.
☐ We did not finish task(s) in time.	☐ I did not transfer all important information.
☐ We did not achieve the goal(s).	☐ We detected a lot of/could not eliminate defect(s).
☐ We did not solve important problem(s).	☐ Comprehension problems/misunderstandings occurred.
☐ We did not work productively.	☐ Technical problem(s) disturbed the session.
☐ other:	

9. Please use this field for further comments?

<u>Note:</u>

* By collaborative software development we mean any development (coding, testing, reviewing, etc.), where two or more developers work jointly, concurrently and simultaneously on the same code base or artifact(s).

C. Leitfäden

Reflexionsleitfaden

---

1. **How did the session go?**

    - What worked well?
    - What did not work?
    - Did you think of something else since you completed the questionnaire?
    - Anything you found special/that striked you?
        - inhaltlich:
        - sozial:
        - technisch:
    - Technically: do you have other requirements for Saros?

2. **Question about General Questions:**

3. **Questions about Upcoming Session Questionnaire:**

4. **Questions about Most Recent Questionnaire :**

5. **My Impressions**: I had the impression that … nur wenn fundiert an video!

6. **Would you like to do something differently next time?**

7. **Reflexion zusammenfassen:**

    1. Things that did not work:
    2. Things that worked well:
    3. Things to do differently next time:

8. **Schwerpunkte in diesem Gespräch waren:**

Leitfaden abschließende Erhebung als teilstrukturiertes Interview

Verbessern des Entwicklungs- sowie des Etablierungsprozesses auf Basis der Bewertungen durch die Beteiligten, d.h. ich möchte wissen, was den Entwicklern im DPP-Projekt wichtig ist und wie ich meinen Etablierungsprozess dahingehend verbessern kann. Dazu erforsche ich im Interview die Ziele der Entwickler und stelle Fragen über deren Bewertungskriterien für Erfolg. Dann vergleiche ich diese Informationen mit den Beobachtungen aus der Sitzung. Daraus schließe ich, ob es sich aus Sicht der Entwickler um erfolgreiche Sitzungen handelt oder was man daran ihrer Meinung nach verbessern könnte.

GQM Template zum Ziel:

Analyze:	Entwicklungs- und Etablierungsprozess
For the purpose of:	zur Verbesserung des Entwicklungs- und Etablierungsprozesses
With respect to:	zur Erhöhung der Nützlichkeit von DPP für das inhaltliche Vorankommen (Projekt) der Entwickler
From the viewpoint of:	aus Sicht der Beteiligten (auch des Beobachters?)
In the context of:	im Kontext der Sitzungen Wien und Bangalore

Ablaufskizze des Interviews:

1. **Have you talked to someone else about your experience in your CSD sessions?** Maybe a colleague or a friend? Someone who is not part of the process?
   Zweck: leicht zu beantwortende Einstiegsfrage + um herauszufinden, was dem Entwickler wichtig ist (darüber hat er dann auch mit anderen gesprochen)
   1a. How did you like your CSD sessions so far? Why?
   Zweck: Alternative zu 1., wenn da nein als Antwort kommt.
2. **Did you achieve your session goals as you imagined you would?**
   - Which were your goals?
   - How did you achieve them? oder Why didn't you achieve them?
   - Are there goals on other levels as well (specific session goals=fix bug, project goals=information transfer or process goals=good knowledge transfer)?

   Zweck: Welche Ziele (Sitzungsziele – jedes Mal neu, Projektziele – „Moving Targets" auch von Dritten abhängig, DPP-Ziele – sitzungsübergreifend) sind vorhanden und wurden diese erfüllt? Wenn sie nicht erfüllt wurden, kann man mit einem der beobachteten Phänomene fortfahren.
   2a. With DPP is it how you imagined it would be? Why?
   Zweck: Alternative zu 2., wenn dort „hatte keine Ziele" als Antwort kommt.
3. **How did you realize that you have (not) achieved your goal(s)?**
   Zweck: Bewertungskriterien für Ziele der Entwickler erfragen. Welche Kriterien sprechen für eine erfolgreiche DPP Sitzung? Haben sie noch dieselben Kriterien, oder sind neue Kriterien aus der Erfahrung erwachsen? Wurden die Kriterien untereinander besprochen? Warum (nicht)? Waren das sinnvolle Kriterien? Werden alle Kriterien bei der Bewertung einbezogen, warum (nicht)?
   3a. How did you realize that it was not what you imagined?
   Zweck: Alternative zu 3., wenn vorher „keine Ziele" als Antwort kommt.
4. **What have you done so far to achieve your goals? Why?**
   Zweck: Wie haben sich die Entwickler vorbereitet. Unausgesprochene Erwartungen aufdecken, falls welche vorhanden sind. Problemlösungsverhalten erklären lassen. Neue Ideen für die Lösung der Probleme entdecken.
   4a What have you done so far to make it more agreeable for you?
   Zweck: Alternative zu 4., wenn vorher „keine Ziele" als Antwort kommt.

D. Saros-Funktionalitäten mit Screenshot

1a) Saros Views:

View	Symbol	Erklärung
Roster		Im Roster können Kontakte und die Verbindung zum Kommunikationsserver verwaltet werden.
Shared Project Session		In der Shared Project Session werden alle Aktivitäten verwaltet, die mit einer aktiven Saros-Sitzung verbunden sind.

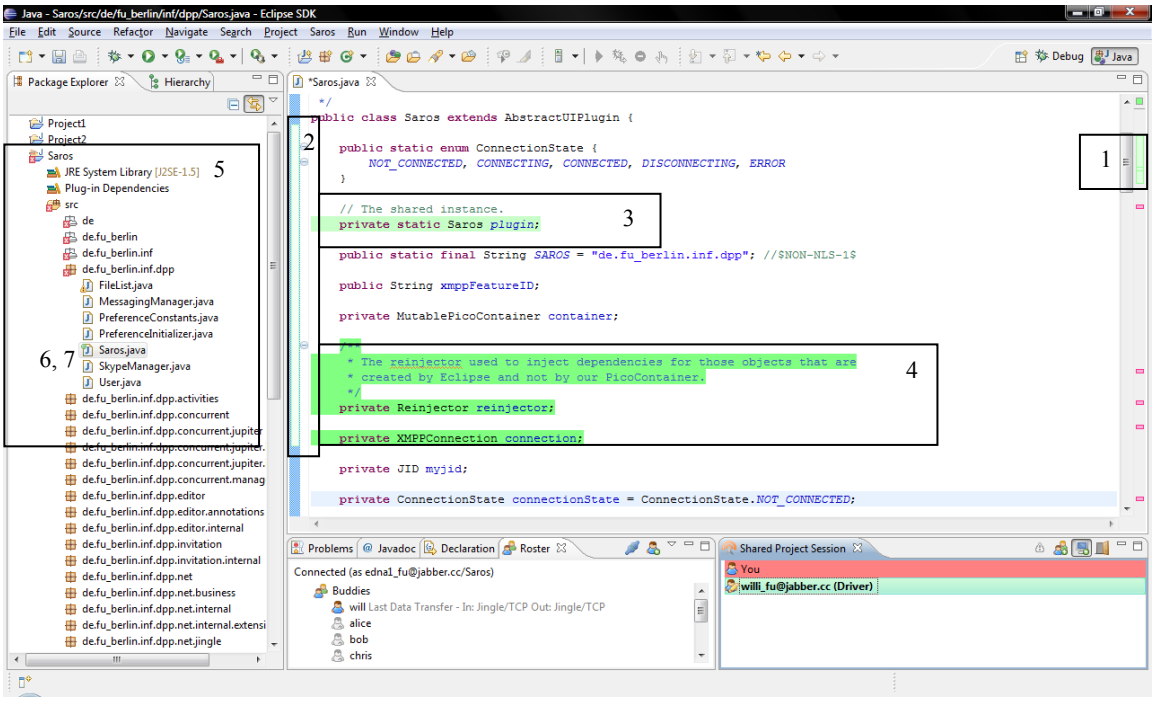
1b) Saros-Funktionalitäten

Roster	Symbol	Erklärung
Mit dem Server verbinden		Mit diesem Knopf kann man sich über das Internet mit dem gemeinsamen Kommunikationsserver verbinden.
Neuen Kontakt hinzufügen		Über diesen Knopf kann man einen Dialog öffnen, der den ersten Kontakt zu einem anderen Buddy herstellt und diesen dann dauerhaft in die Buddiesliste übernimmt.
Shared Project Session	**Symbol**	**Erklärung**
Zur Projektsynchronisation einladen		Über den Einladungsdialog, der daraufhin erscheint, kann man auswählen, mit welchen anderen Benutzern man Informationen teilen möchte.
Verfolgermodus		Wenn man diese Funktion aktiviert, werden alle weiteren Aktionen des Drivers (Öffnen von Dateien, Änderungen, etc.) verfolgt.
Single-Driver Mode		Der Host kann sich oder einem anderen die Driver-Rolle zuweisen und der Driver hat die Schreibrechte.
Multi-Driver Mode		Der Host kann sich und anderen die Driver-Rolle gleichzeitig zuweisen und paralleles Editieren aller Driver wird unterstützt.
Jump to position of selected user		Über diese Funktionalität wird man einmalig an die Stelle geführt (Datei öffnet sich), an der sich der Cursor des gewählten Benutzers gerade befindet.
Watchdog		Inkonsistenzanzeige. Diese Anzeige erscheint bei anderen Benutzern, wenn der Zustand der Dateien nicht dem des Hosts entspricht.
Sitzung verlassen		Mit diesem Knopf kann man jederzeit die aktuelle Saros-Sitzung verlassen.

1c) Saros Awareness Funktionalitäten

Viewport Annotations	Nr.	**Erklärung** (siehe Screenshot)
Viewport Annotation	1.	Rechts im Editorfenster, relativer Sichtbereich der Datei des Drivers und des zu Verfolgenden im Verhältnis zur Gesamtgröße der Datei wird in der Viewport Annotation Bar angezeigt.
Detail Viewport Annotation	2.	Links im Editorfenster, zeigt die Zeilen, die sich der Verfolgte gerade ansieht; sieht man den Detail Viewport nicht, so sieht man nicht alle Zeilen.
Contribution Annotation	3.	Die letzten X Zeichen, die der andere geschrieben hat.
Selection Annotation	4.	Was der andere gerade markiert hat und seine Cursorposition (in 1.).

Package Explorer (links)	**Nr.**	**Erklärung** (siehe Screenshot)
Shared project decoration	5.	Projekt, das geteilt wurde, hat ein Sendesymbol an der Datei.
Shared project file decoration	6.	Grüner Punkt, verfolgter Driver ist in dieser Datei.
Shared project file decoration	7.	Gelber Punkt, Datei ist bei Verfolger und Driver offen.



Screenshot von Saros (Version 9.7.31 r1265) in Eclipse (Version 3.4.1) mit Roster und Shared Projekt Session Views und Markierungen 1-7 für Awareness Funktionalitäten

E. Anwendungsfälle

Anwendungsfall: DDP-Walkthrough (Review)

Primärakteur: Autor des Codes (kurz Autor)

Stakeholder und Interessen:

1. Reviewer: prüft die semantische Korrektheit des Codes und möchte dafür einen Überblick über den Status der Entwicklung.
2. Autor: möchte eine Bestätigung, dass der bisherige Code der vorläufigen FAD/TAD entspricht und Entscheidungen über unklare Entwurfsentscheidungen vom Reviewer.

Vorbedingungen:

1. Die Entwickler haben die technische Infrastruktur etabliert, und
2. alle nötigen Dateien (Quellcode und Spezifikationsdokumente) liegen allen Sitzungsteilnehmern vor.

Nachbedingungen (bei Erfolg):

1. Der Reviewer ist über den Status der Entwicklung informiert.
2. Der Autor hat eine Bestätigung für die Korrektheit des Verhaltens seines bisherigen Codes, und alle weiteren Entwurfsentscheidungen wurden geklärt.

Nachbedingungen (bei Misserfolg):

1. Der Reviewer hat noch keinen vollständigen Überblick über das Verhalten des Codes.
2. Der Autor hat nur teilweise Bestätigungen für die Korrektheit des Verhaltens seines bisherigen Codes, und/oder nicht alle Entwurfsentscheidungen konnten geklärt werden.

Standardablauf:

1. Der Autor oder der Reviewer lädt zur Synchronisation des Projektes ein.
2. Die Driver-Rolle geht an den Autor über, falls er sie nicht schon hat, und der Reviewer schaltet den Verfolgermodus an.
3. Der Autor stellt allgemeine Entwurfsfragen oder öffnet eine Datei und macht einen Vorschlag zur Umsetzung, indem er das Verhalten seines Codes vorstellt.
4. Der Reviewer öffnet die Spezifikationsdokumente, und die beiden diskutieren anhand des Codes und der Spezifikationsdokumente über die Entwurfsentscheidungen.
5. Der Reviewer stimmt dem Vorschlag zu oder trifft eine andere Entwurfsentscheidung.
6. Die Sitzung wird beendet oder das Review wird fortgesetzt (weiter bei Schritt 3).

Erweiterungen/alternative Abläufe:

4a. Der Reviewer hat die Frage noch nicht verstanden und stellt Rückfragen zur Klärung.
4b. Der Reviewer liest im Spezifikationsdokument nach, um die Frage zu erörtern.
5a. Der Reviewer kann vorläufig noch keine Entwurfsentscheidung treffen und macht sich eine Markierung in den Spezifikationsdokumenten an der Stelle, die noch unklar ist.

Häufigkeit des Auftretens: selten (3/16)

Anwendungsfall Pair-Developing

Primärakteur: Domainerfahrener Entwickler (Experte)

Stakeholder und Interessen:

3. Experte: möchte den Status der Entwicklung besprechen und weitere Sitzungsaufgaben (die Weiterentwicklung der Funktionalitäten) erledigen.
4. Zweiter Entwickler (Novize): möchte die Sitzungsaufgabe erledigen.

Vorbedingungen:

3. Die Entwickler haben die technische Infrastruktur etabliert, und
4. alle nötigen Dateien (Quellcode) liegen allen Sitzungsteilnehmern vor.

Nachbedingungen (bei Erfolg):

3. Die geplanten Sitzungsaufgaben konnten erledigt werden.

Nachbedingungen (bei Misserfolg):

3. Die geplanten Sitzungsaufgaben konnten nur teilweise erledigt werden und müssen auf die nächste Sitzung verschoben werden.

Standardablauf:

7. Der Experte oder der Novize lädt zur Synchronisation des Projektes ein.
8. Der Experte und der Novize besprechen den aktuellen Status des Quellcodes (die Driver-Rolle geht an denjenigen, der die letzten Änderungen vorgenommen hat).
9. Der Experte fängt an, den Quellcode gemäß der geplanten Sitzungsaufgabe zu verändern oder zu erweitern.
10. Der Novize stellt Rückfragen, macht den Experten auf Fehler aufmerksam oder macht Änderungsvorschläge.
11. Der Experte oder der Novize fahren fort zu kodieren und wechseln sich dabei als Driver ab, bis die Aufgabe beendet ist.
12. Der Experte plant das weitere Vorgehen, fährt mit einer weiteren Sitzungsaufgabe fort oder beendet die Sitzung.

Erweiterungen/alternative Abläufe:

4a. Es treten unvorhergesehene Probleme auf, die die Entwickler gemeinsam zu lösen versuchen.

5b. Die Entwickler starten einen Testlauf und versuchen, anhand des Testberichts die Ursache für Probleme zu finden.

6a. Die Entwickler können die Aufgabe in der geplanten Zeit nicht beenden und müssen die Beendigung der Aufgabe auf die nächste Sitzung verschieben.

Häufigkeit des Auftretens: oft (9/16)

F. Detaillierte Beschreibung der Sitzungsverläufe

Chronologische Beschreibung der Ereignisse in der 7. Sitzung (Sitzungstyp Review)

In der Beschreibung werden die Ereignisse chronologisch aufgelistet. Die Zeiträume (Zeitpunkt), das aktive Fenster (Fokus) und der aktive Entwickler (Akteur) werden getrennt aufgeführt. Die grauen Markierungen zeigen alle Phasen an, in denen Saros eingesetzt wurde bzw. Aktionen, die inhaltlich zusammengehören.

Zeit	Fokus	Akteur	Aktion und weitere Erklärungen
11:03	Saros	E1	Projektsynchronisation, Entwickler1 (E1) nennt Sitzungsziel: *„get short intro into code“* und gibt Anweisung über Vorgehen: *„open every class you want and you can highlight“*
		E3	Entwickler3 (E3) nennt Vorgehen und Status der Entwicklung: *„I will give you status of how far we are and where we are stuck“*, stellt allgemeine Entwurfsfragen über Typbezeichnung, mögliche Fehler in FAD und erwünschtes Verhalten
	FAD	E1, E3	Dafür öffnet jeder bei sich das FAD
		E1, E3	Entwickler klären allgemeine Ressourcennutzung (Server)
11:10	Saros	E3	Entwickler3 bekommt Driver-Rolle
		E3	Öffnet Datei 1 (D1), erklärt Verhalten der Klasse
		E3	Öffnet D2, erklärt Zusammenhang zwischen den Klassen, erklärt innerhalb D2 Methoden und Parameter nacheinander
		E1	Stellt Rückfrage über Verwendung eines Parameters an einer Schnittstelle
		E3	Öffnet zwei weitere Dateien D3+D4 zur Erklärung auf die Rückfrage
		E3	Öffnet zwei weitere Dateien D5+D6. Macht mit Erklärung über Methoden weiter, wechselt zwischen D5 und D6 hin und her, nutzt den Cursor, um Stellen zu markieren, über die er spricht
11:20		E3	Stellt eine allgemeine Entwurfsfrage (wann eine bestimmte Variable im Ablauf des Programms zur Verfügung stehen soll)
		E1	Denkt nach und stellt Rückfragen
		E1, E3	Diskutieren kurz über eine sinnvolle Anpassung (bzgl. der Variable)
		E1	Entscheidet, die Anpassung an dieser Stelle wegzulassen
	FAD	E1	Abschließend öffnet E1 die FAD und nimmt kleine Änderungen diesbezüglich vor
11:22	Saros	E3	Macht mit seiner Präsentation weiter, geht in D5 und erklärt seine Änderungen am Code (verwendet den Cursor, um Stellen zu markieren über die er spricht)
		E3	Öffnet neue Datei D7, will etwas erklären. Entscheidet sich um, weil er vergessen hat, die Herkunft eines Parameters zu erklären. Er sucht nach einer Datei, um es anhand eines Beispiels zu erklären
	FAD	E1	Schaut sich kurz die FAD an, während E3 sucht
	Saros	E3	Hat weitere drei Dateien D8, D9 und D10 geöffnet, sucht Startpunkt für das Beispiel
		E3	Entscheidet sich dafür, vor dem Beispiel noch etwas in F10 zu erklären
		E3	Erklärt weiter in D10
		E3	Öffnet weitere Datei D11 zur Erklärung von Parameter aus D10
		E3	Erklärt in D11 sein Beispielszenario und fragt nach erwartetem Verhalten (die weiteren 5 Schritte wurden unternommen, um die offene Frage zu klären).
		E1	Denkt nach und stellt Rückfragen

		E3	Wechselt zu D10 und sucht dort nach Informationen zur weiteren Klärung
11:35		E3	Macht zusätzlich einen Testanruf, um weitere Informationen zur Klärung der Frage zu haben (davon kann E1 nichts sehen)
		E1	Stellt Rückfragen über den Testanruf
	FAD	E1	Schaut kurz in die FAD und verschiebt schließlich die Klärung der offenen Frage. Macht sich Markierungen in die FAD an den Stellen, die er sich später alleine noch mal anschauen muss (es bleiben zwei offene Punkte)
11:40		E3	Wechselt das Thema und stellt allgemeine Frage, welchen (VoIP) Softclient E1 für Tests verwendet
		E1	Gibt Auskunft darüber und sagt wo die Software zu finden ist
11:49		E1, E3	Beenden das Review und planen eine weitere Session für die offenen Punkte

Chronologischer Ablauf der 7. Sitzung (einer Review-Sitzung) zwischen Entwickler1 (E1) und Entwickler3 (E3) im DPP-Projekt

Chronologische Beschreibung der 5. Sitzung (Sitzungstyp Pair-Developing)

Die Tabelle zeigt eine chronologische Abfolge der Ereignisse und zusätzliche Angaben über den aktiven Entwickler (Akteur) und seine Aktivitäten (Aktion).

Zeit und Aktivität	Ak-teur	Aktion und weitere Erklärungen
10:46	E1	Entwickler1 (E1) ist Host. Während der Projektsynchronisation erklärt E1 das Projektziel: *„to implement a voice recording application“* und beschreibt ganz kurz die Hauptfunktionalitäten (Knöpfe), die durch die Anwendung unterstützt werden sollen.
10:59 Code Prä-sentation	E1	E1 ist Driver. Er erklärt E2 die Modulstruktur (indem er im Package Explorer durch die Dateien geht).
	E1	E1öffnet eine Datei D1, nennt die Methoden, die darin sind, und was die wichtigsten Funktionen darin sind (durch Scrollen und Bewegungen mit dem Cursor).
	E1	E1 öffnet weitere Dateien D2-D5 und erklärt chronologisch das Verhalten des Programms, teilweise zeigt er dabei auch auf bestimmte Parameter und springt an andere Stellen im Code, wo diese implementiert sind.
	E1	Danach geht E1 an die Stellen, die noch bearbeitet werden müssen, erklärt, was dort zu tun ist und schreibt ein „TODO“ an diese Stellen.
	E1	Im letzten Teil der einleitenden Erklärung zeigt E1 in D2 Stellen, die noch überarbeitet werden müssen.
11:06 Planung	E1	E1 nennt die offenen Aufgaben („TODOs“, Fehlerbehandlung und Erstellung von FAD/TAD). Er fragt E2 nach dessen Verfügbarkeit und teilt die Aufgaben ein. E1 ist für die Erstellung des Anforderungsdokumentes bis zur folgenden Woche, E2 und E3 sind für die Erstellung der FAD/TAD verantwortlich.
	E2	E2 stellt eine Frage zu der Methode in D2, die noch überarbeitet werden soll.
	E1	E1 teilt wie vorher mit, dass er sich nicht sicher ist, wie die Methode richtig implementiert werden kann.
	E2	E2 macht einen Lösungsvorschlag.
	E1, E2	Sie diskutieren kurz über den Vorschlag und Entwickler1 sagt, dass er alle Informationen, die er darüber hat, ins Anforderungsdokument schreiben wird. Außerdem plant er, deswegen noch einmal mit dem verantwortlichen Produkt Manager (technischer Projektleiter) zu sprechen.
	E1	E1 erwähnt, dass die Deadline für die Beendigung des Projektes (siehe erste Aktion) im Sommer sein soll.
	E2	E2 fragt nach globalen Parametern.
	E1	E1 nennt sie kurz und schreibt weitere „TODOs“ in den Code an der Stelle, an der globale *Identifier* enthalten sein sollen.
	E1	E1 schließt die Planungsphase ab, indem er Aufgaben für die nächste Sitzung plant. Dazu fragt er E2 nach dem Status der Aufgaben von E3 und beschließt, als nächstes eine Review-Sitzung mit E2 und E3 durchzuführen.
11:20 Implemen-tierung	E1	E1 hatte vorgehabt, die Sitzung an dieser Stelle zu beenden. Da aber noch etwas Zeit bis zur Mittagspause war, entschied er sich, eines der „TODOs“ zu bearbeiten. Er wählte die Stelle in D2, die noch überarbeitet werden sollte.
	E1	E1 ist weiterhin Driver. Er nimmt einige Änderungen an einer der Methoden vor und notiert darin ein weiteres „TODO“ (eine Check-Bedingung, die noch fehlt).
	E2	E2 fragt nach dem Ursprung von Parametern in der Methode.
	E1	E1 wechselt in eine andere Datei D6 und zeigt mit dem Cursor auf die Stelle. Ohne weitere Erklärung wechselt er zurück zu D2.

	E1	E1 macht weitere Änderungen in D2.
	E1	E1 erklärt welche Funktionalitäten er gerne hätte und berät sich mit E2 darüber welche Datenstruktur er für die Verwaltung der Daten verwenden sollte.
	E2	E2 teilt seine Meinung darüber mit, und sie einigen sich auf den Vorschlag von E1.
	E1	E1 denkt kurz darüber nach, wo die Daten abgelegt werden sollten, lässt sie durch Eclipse automatisch generieren und erzeugt drei weitere Methoden damit.
	E1, E2	E1 und E2 diskutieren die Deklaration der neuen Methoden.
11:29	E1	E1 fordert E2 auf, die Implementierung der Methoden zu übernehmen und gibt ihm dafür die Driver-Rolle.
	E2	E2 implementiert die drei Methoden. Diese sind jedoch wegen technischer Probleme für E1 nicht sichtbar.
		Wegen technischer Probleme (Inkonsistenzen) muss die Saros-Sitzung neugestartet werden. Die Änderungen von E2 gehen dabei verloren.
11:40	E2	E2 ist jetzt wieder Driver und implementiert die erste Methode. Danach fragt er E1, ob er weiter machen soll.
11:41	E1	E1 übernimmt die Driver-Rolle wieder, um die zweite Methode zu schreiben. Dabei fragt er kurz nach, ob er eine weitere Check-Bedingung einfügen soll.
11:43	E2	E2 bekommt die Driver-Rolle wieder und implementiert die dritte Methode.
11:43	E1	E1 beendet die Sitzung, indem er die Driver-Rolle zurücknimmt, das Projekt abspeichert und letzte „TODOs“ an die Stellen schreibt, an denen er die neuen Methoden verwenden will.

Chronologischer Ablauf der 5. Sitzung (einer Pair-Developing-Sitzung) zwischen Entwickler1 (E1) und Entwickler2 (E2) im DPP-Projekt

Abbildungsverzeichnis

Abbildung 2.1 a) Location, Nimbus und Focus nach Rodden b) „Mode Overlap 1“ des Awarenessmodells von Rodden 10
Abbildung 2.2 Arten von Awareness in kollaborativer Zusammenarbeit von Gutwin et al. [GGR96] 10
Abbildung 2.3 Formalitätsgrade von Durchsichten nach Wiegers [Wie01] 14
Abbildung 3.1 CAR-Prozessmodell nach Davison et al. [DMK04] 19
Abbildung 3.2 GQM-Modell nach Basili et al. [BCR94] 22
Abbildung 4.1 Das Saros-Team und die Positionen der Teammitglieder 29
Abbildung 4.2 Technische Infrastruktur der DPP-Sitzungen bei Teles 30
Abbildung 4.3 Aufteilung und Ablauf der Arbeitszyklen des Forschungsprozesses innerhalb des Etablierungsprozesses 41
Abbildung 5.1 Angaben von Entwickler2 über die Prioritäten die Sitzungsaufgaben pünktlich zu erledigen und alle wichtigen Informationen zu erhalten (1 = sehr wichtig bis 6 = am wenigsten wichtig) 60
Abbildung 5.2 Angaben der Entwickler im Fragebogen US über die Priorität dafür, dass keine technischen Probleme die Sitzung behindern (US-kurz Frage 7, 1= sehr wichtig bis 6 = am wenigsten wichtig) 71
Abbildung 5.3 Angaben im Fragebogen US zu Frage 7 von Entwickler2 über die höchsten Prioritäten für den Erfolg der Sitzung (mit Mehrfachangaben) 75
Abbildung 5.4 Zeitliche Anteile von Verzögerungen (durch Verspätungen und lange Vorbereitungsphasen) und von produktiver Arbeitszeit in den DPP-Sitzungen 92
Abbildung 5.5 Angaben der Entwickler im Fragebogen nach den Sitzungen (MRS, Frage 8) über den Erfolg der Sitzung 93
Abbildung 5.6 a) Angaben von Entwickler1 und b) Entwickler2 im Fragebogen nach der Sitzung (MRS Frage 3) darüber wie hilfreich DPP in der Sitzung war 93

Tabellenverzeichnis

Tabelle 2.1 Elemente von Workspace Awareness [GGR96] 11
Tabelle 4.1 Verwendete Software für die technischen Infrastruktur der DPP-Sitzungen im Etablierungsprozess 31
Tabelle 4.2 Persönliche Informationen über am DPP-Projekt beteiligte Teles-Entwickler 31
Tabelle 4.3 Angaben der Entwickler im Fragebogen General Questions darüber, was sie generell über DPP denken 32
Tabelle 4.4 Angaben der Entwickler im Fragebogen General Questions über die Kriterien einer erfolgreichen Sitzung 33
Tabelle 4.5 Angaben der Entwickler im Fragebogen General Questions über die Kriterien einer erfolglosen Sitzung 33
Tabelle 4.6 Allgemeine Ziele des Teams und der einzelnen Entwickler im DDP-Projekt 33
Tabelle 4.7 Aufteilung der Entwicklungsaufgaben zwischen den Entwicklern im DPP-Projekt .. 35
Tabelle 4.8 Geplanter Ablauf an Sitzungstagen bis zur 5. Sitzung, inkl. Reflexion, Fragebogen Upcoming Session (US) und Fragebogen Most Recent Session (MRS) 38
Tabelle 4.9 Änderungen im Ablauf an Sitzungstagen während des Etablierungsprozesses 38
Tabelle 4.10 Geplanter Ablauf an Sitzungstagen ab der 12. Sitzung, inkl. Reflexion, Fragebogen Upcoming Session (US) und Fragebogen Most Recent Session (MRS) 38
Tabelle 4.11 Übersicht über Arbeitszyklen, DPP-Sitzungen und Reflexionen im DPP-Projekt ... 42
Tabelle 4.12 Maßnahmen in der Pilotphase 42
Tabelle 4.13 Maßnahmen im ersten Arbeitszyklus 44
Tabelle 4.14 Maßnahmen im zweiten Arbeitszyklus 48
Tabelle 4.15 Maßnahmen im dritten Arbeitszyklus 50
Tabelle 4.16 Maßnahmen im vierten Arbeitszyklus 52
Tabelle 5.1 Erklärungsversuche von Entwickler3 bei einer Codepräsentation in der 7. Sitzung .. 61
Tabelle 5.2 Informationen zu Sitzungsausfällen, die durch Entwickler1 verursacht wurden 64
Tabelle 5.3 Zusammenfassung der Sitzungsausfälle verursacht durch Entwickler1 wegen dringenderer Aufgaben 65
Tabelle 5.4 Ausfälle von Sitzungen, die durch Entwickler2 verursacht wurden 65
Tabelle 5.5 Ausfälle von Sitzungen, die durch Entwickler3 verursacht wurden 66
Tabelle 5.6 Zusammenfassung der Ausfälle von Entwickler2 oder Entwickler3 67
Tabelle 5.7 Ausfälle von Sitzungen, die durch Entwickler3 verursacht wurden 67
Tabelle 5.8 Zusammenfassung der Ausfälle von Entwickler2 oder Entwickler3 68
Tabelle 5.9 Ausfall des produktiven Einsatzes von DPP wegen fehlender Freigabe (NDA) für die Betreuung 68
Tabelle 5.10 Absage der Betreuung für die 4. Sitzung 69
Tabelle 5.11 Zusammenfassung der Sitzungsausfälle wegen fehlender Betreuung 70
Tabelle 5.12 Absage der 14. Sitzung von Entwickler1 wegen technischer Probleme 72
Tabelle 5.13 Zusammenfassung der Sitzungsausfälle wegen fehlender Voraussetzungen oder technischer Probleme 72
Tabelle 5.14 Übersicht über Ausfälle von Sitzungen oder ausgefallene Entwickler 73
Tabelle 5.15 Verspätungen von Entwickler2 zu Sitzungen 75
Tabelle 5.16 Zusammenfassung von Verspätungen von Entwickler2 zu Sitzungen 79
Tabelle 5.17 Verspätungen der Entwickler zu Reflexionen 79
Tabelle 5.18 Zusammenfassung der Verspätungen der Entwickler zu Reflexionen 80
Tabelle 5.19 Verspätungen von Sitzungen durch Reflexionen 81
Tabelle 5.20 Zusammenfassung der Verspätungen von Sitzungen durch vorherige Reflexionen. 82
Tabelle 5.21 Übersicht über Verspätungen von Entwicklern zu Sitzungen, zu Reflexionen und durch Reflexion im Etablierungsprozess 83
Tabelle 5.22 Übersicht über Formen von fehlender Awareness im Etablierungsprozess 90