Anforderungen im Projekt DPP (Stand: 28.10.2008)

Um im Rahmen der Diplomarbeit *DPPIV* eine analytische Qualitätssicherung im Projekt DPP betreiben zu können, müssen zunächst die zugrunde liegenden Anforderungen geklärt werden.

Hierfür wurden die in den vorhanden Dokumenten aufgestellten Anforderungen zunächst zusammengetragen und neu bewertet. Diese Neubewertung war aufgrund von folgenden Tatsachen nötig:

- Anforderungen aus *DPPI* und *DPPII* sind teilweise nicht mehr gültig (z.B. wegen Multi-Driver Funktionalität), wurden im Rahmen von DPPIII aber nicht modifiziert
- teilweise wurden Anforderungen zu ungenau beschrieben oder fehlen komplett.

Ziel dieses Dokumentes ist es durch eine Art „Restaurierung“ der Anforderungen dem Softwareprojekt eine neue Produktdefinition zu geben.

Die Anforderungen werden durch zwei sich ergänzende Notationen beschrieben:

1. Anforderungskatalog
2. Use Cases.

Anforderungskatalog

In diesem Anforderungskatalog werden die funktionalen sowie die nichtfunktionalen Anforderungen jeweils getrennt voneinander aufgestellt.

Funktionale Anforderungen

ID	*Titel*	*Beschreibung*
FR1	Kollaboration	Das Werkzeug muss mehreren verteilten Entwicklern (auch mehr als 2) ein gemeinsames Arbeiten am Quellcode innerhalb eines Projektes ermöglichen. Alle Entwickler müssen sowohl als *Driver* als auch als *Observer* an der Programmiersitzung teilnehmen können.
FR2	Kommunikation	Es muss eine Kommunikation zwischen den Teilnehmern der Programmiersitzung möglich sein. Als Mindestanforderung muss diese Kommunikation über Textnachrichten geschehen können. Dabei soll es möglich sein, eine Nachricht an alle oder nur an einen bestimmten Teilnehmer zu schicken. Die Kommunikation über Sprache und/oder Video muss *nicht* durch das Werkzeug selber geleistet werden, hierfür können externe Programme herangezogen werden.
FR4	*Parallelität auf Sichtebene*	Den Observern ist es nicht nur möglich andere Programme neben der Entwicklungsumgebung auszuführen (*Parallelität auf Programmebene*) sondern auch innerhalb der Entwicklungsumgebung von dem Aufmerksamkeitsbereich des/der Driver abzuweichen und eigenständig Dateien eines Projekts anzuschauen.
FR5	*Parallelität auf Schreibebene*	Das Werkzeug muss es ermöglichen, dass es zur gleichen Zeit mehrere Driver geben kann, d.h. es kann mehr als ein Entwickler Schreibzugriff auf den Quellcode besitzen. Die Driver sollen einfache Editieroperationen (Einfügen/Löschen) blockierungsfrei durchführen können. Für Dateioperationen und komplexere Editieroperationen (Refactoring) können temporär einzelne Dateien gesperrt werden (exklusives Schreibrecht). Die Die Undo-Funktion (der eigenen Änderungen) muss weiterhin funktionieren.
FR6	*Awareness*	Es muss zu jeder Zeit visualisiert werden, wer gerade an welcher Datei arbeitet (z.B. durch Markierung in Dateiliste). Es muss die aktuelle Cursorposition der Driver übertragen werden und die jeweiligen Sichtbereiche und Änderungen dieser visuell erkennbar sein (z.B. durch

ID	*Titel*	*Beschreibung*
		global eindeutige Hintergrundfarben). Die Observer sollen die Möglichkeit besitzen, Textpassagen zu selektieren und dies soll für die Driver sichtbar sein.
FR7	Verfolgermodus	Es muss den Observern möglich sein, die Veränderungen von den Drivern am Quellcode automatisch zu beobachten, d.h. es wird dem Sichtbereich eines Drivers gefolgt. Hierzu kann man den zu beobachtenden Driver auswählen. Der Verfolgermodus soll vom Observer selbständig aktiviert und deaktiviert werden können.
FR8	Rollenwechsel	Der die Programmiersitzung initiierende Entwickler hat zu jeder Zeit die Möglichkeit den Teilnehmern die Driver-Rolle zu geben und wieder zu entziehen.
FR9	Übersetzbarkeit des Quellcodes	Bei allen Teilnehmern der Programmiersitzung muss der Quellcode genau so übersetzt werden können als wäre es eine normale nicht verteilte Programmiersitzung.
FR10	Projektfreigabe	Als Zugriffskontrolle muss eine Freigabe auf Projektebene möglich sein.
FR11	Auswahl von Programmierpartnern	Es muss eine Liste von bekannten Programmierpartnern angezeigt werden und ob diese gerade für eine Programmiersitzung verfügbar sind.
FR12	Integration in ElectroCodeoGram (ECG)	Es muss ein ECG-Sensor vorhanden sein, der die Rollenwechsel und Veränderungen aufzeichnet. Des Weiteren muss erkennbar sein, wer welche Änderungen durchgeführt hat. Als Mindestanforderung muss aus einem Log von einem Entwickler ersichtlich sein, welche Änderungen am Quellcode von ihm selber und welche Änderungen von anderen kommen.
FR13	Zustandskontrolle	Es muss eine Zustandskontrolle geben die den aktuellen Zustand der Verbindung überwacht. Dem Benutzer soll der Durchsatz (In/Out) der aktuellen Verbindung angezeigt werden.

Nichtfunktionale Anforderungen

ID	*Titel*	*Beschreibung*
NFR1	Zielplattform	Das Werkzeug muss in Eclipse (3.4) unter Windows und Linux mit installiertem JDK (ab 1.5) lauffähig sein.
NFR2	Installation	Die Installation muss über eine Eclipse Updatesite in weniger als 5 Minuten möglich sein.
NFR3	Dokumentation	Es muss Dokumentation für folgende Personengruppen existieren: Entwickler, Administratoren und Benutzer. Alle externen Dokumente müssen in englischer Sprache vorliegen.
NFR4	Dokumentation des Quellcodes	Der Quellcode muss auf Methoden-, Klassen- und Paketebene dokumentiert sein.
NFR5	Initiierung einer Programmiersitzung	Voraussetzung für das Initiieren einer Programmiersitzung ist, dass Saros bei allen Teilnehmer installiert ist, alle Teilnehmer online sind und sich gegenseitig in Saros als Buddies hinzugefügt haben. Die Synchronisation muss effizient sein in Abhängigkeit der Differenz der Dateien. Die gesamte Zeit berechnet sich aus folgenden Laufzeiten: T1 = Checksummen berechnen T2 = Checksummen übertragen T3 = Differenz berechnen T4 = Differenz übertragen Die benötigte Zeit für das Berechnen einer Checksumme und der Differenz soll vergleichbar (asymptotisch) zu denen der gängigen Tools wie rsync sein (wahrscheinlich nlogn). Die Übertragungszeiten hängen von der verfügbaren

ID	*Titel*	*Beschreibung*
		Bandbreite ab, sollte aber effizient sein indem wenn möglich die Daten vor der Übertragung komprimiert werden.
NFR6.1	Netzwerk	Das Werkzeug MUSS in allen Netzwerktopologien eingesetzt werden können, sofern eine Verbindung zum Jabber-Serververbund aufgebaut werden kann.
NFR6.2		Für die Kommunikation zwischen zwei Nutzern *soll* eine Direktverbindung genutzt werden.
NFR6.3		Für den Austausch von Dateien *muss* eine Direktverbindung genutzt werden, sofern diese durch Verwendung der letzten veröffentlichen Stabilen Version der Bibliothek Smack möglich ist. Sollte eine Direktverbindung nicht möglich sein, so muss es eine Rückfallstrategie geben, die trotzdem das Funktionieren des Werkzeugs sicherstellt.
NFR6.4		Es MUSS für jeden Benutzer möglich sein, den Zustand seiner Verbindung bzw. seiner Verbindungen zu den anderen Benutzern einzusehen. Dies beinhaltet: 1.) *(muss)* Direktverbindung oder IBB bzgl. Austausch von Dateien 2.) *(soll)* Direktverbindung oder Transport über Server bzgl. der restlichen Kommunikation 3.) *(muss)* Verstrichene Zeit seit Eintreffen des letzten Datenpakets 4.) *(soll)* Größe oder Anzahl der Daten, die noch auf Versand warten 5.) *(soll)* Durchschnittliches Datenaufkommen in KB pro Sekunde gemittelt über die letzten 60 Sekunden und über die gesamte Sitzung. 6.)*(soll)* Totales Datenaufkommen in MB
NFR6.5		Eine Chatkommunikation *muss* zwischen den Teilnehmern möglich sein, selbst falls der Jabber-Server des Hosts keine Möglichkeit zur Erstellung eines Multi-User-Chats besitzt.
NFR7	Konsistenz	Jeder Teilnehmer der Programmiersitzung muss ein konsistentes Abbild aller Dateien des gemeinsamen Projekt besitzen. Inkonsistenzen müssen innerhalb von 15 Sekunden erkannt und gelöst werden.
NFR8	Konsistenzmodell	Die Nebenläufigkeitskontrolle für den Gruppeneditor hat folgenden Anforderungen zu genügen: • Konvergenz • Kausalitätserhaltung • Intentionserhaltung.
NFR9	Echtzeit	Die lokalen Änderungen müssen verzögerungsfrei wie bei einem Single-Editor durchgeführt werden. Änderungen von entfernten Entwicklern können mit einer Verzögerung von ein paar wenigen Sekunden nachvollzogen werden.
NFR10	Bugtracker	Für das Projekt muss einen Bug/Issuetracker existieren.
NFR11	Öffentliche Mailingliste	Für das Projekt muss eine öffentliche Mailingliste existieren.
NFR12	Projektseite	Für das Projekt muss eine öffentliche Projektseite im Internet existieren.

Use Cases

In diesem Kapitel werden Anwendungsfälle von *Saros* beschrieben. Diese ergänzen nicht nur den Anforderungskatalog für eine genauere Produktdefinition sondern liefern auch mögliche Testszenarien.

Das folgende Diagramm gibt lediglich eine Übersicht über die Use-Cases. Die eigentlichen Use-Cases werden danach in einer halb-formalen Art und Weise in Textform beschrieben.

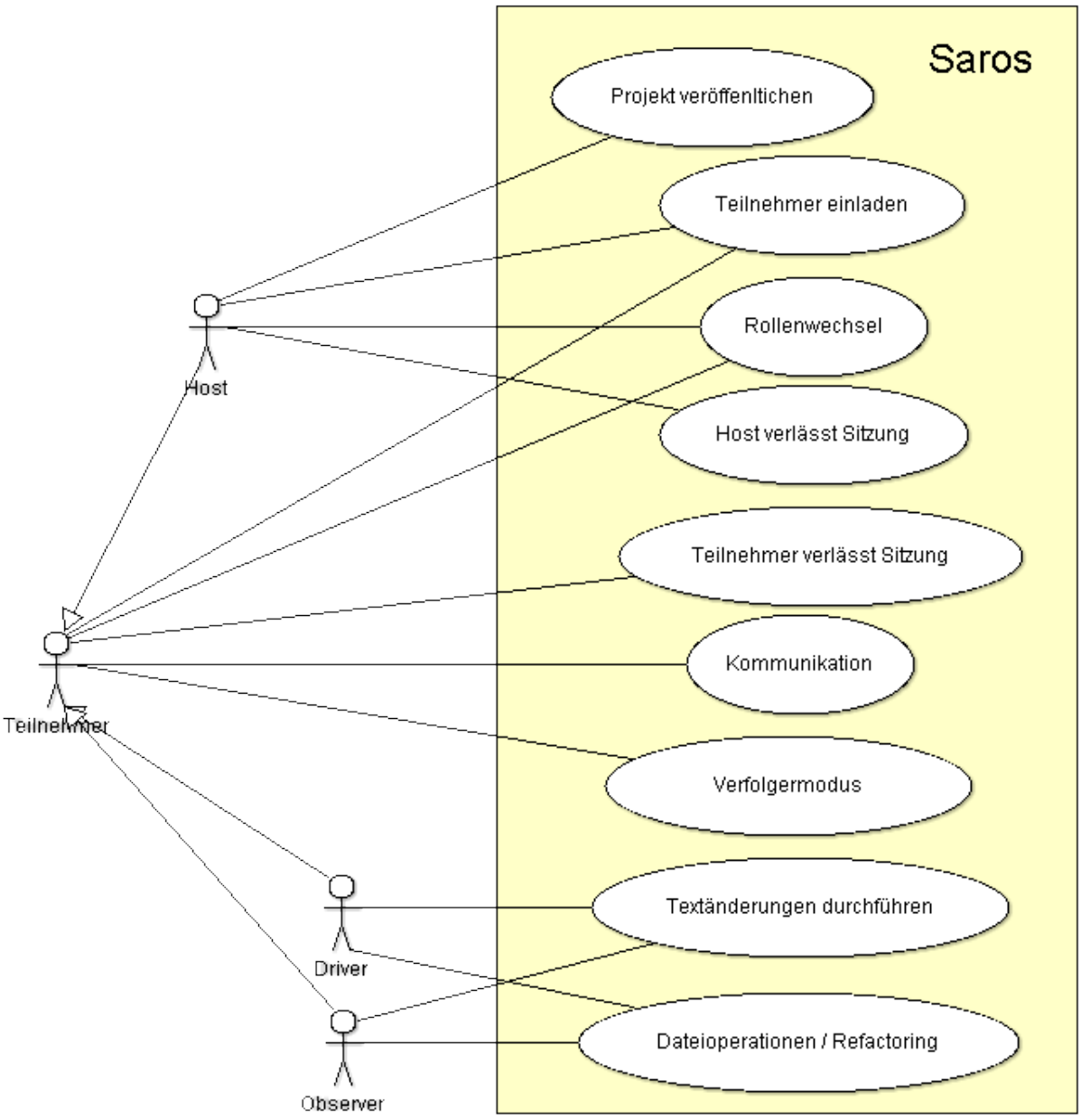

Abbildung: Use Cases Kontextdiagramm

Use Case 1

Name: **Projekt veröffentlichen**

Primärakteur: Host

Stakeholder und Interessen:

- Host: Möchte eine verteilte Programmiersitzung erstellen

Vorbedingungen:

- Saros muss korrekt installiert sein
- Host muss mit Kommunikationsnetz verbunden sein (Jabber)

Nachbedingungen:

- verteilte Programmiersitzung ist erstellt
- Host hat zunächst die Rolle Driver und wird im View *Shared Project Session* als solcher angezeigt
- die Möglichkeit ein weiteres Projekt zu veröffentlichen im Kontextmenu deaktiviert (ausgegraut)

Standardablauf:

1. Host wählt im Kontextmenu eines Projektes „Share Project“ aus
2. Einladungs-Dialog wird angezeigt (siehe Use Case *Teilnehmer einladen*)

Erweiterungen / alternative Abläufe: keine

spezielle Anforderungen: keine

Häufigkeit des Auftretens: mittel

Use Case 2

Name: **Teilnehmer einladen**

Primärakteur: Host

Stakeholder und Interessen:

- Host: Möchte Teilnehmer zu einer verteilten Programmiersitzung einladen
- Teilnehmer: möchten an der Programmiersitzung teilnehmen

Vorbedingungen:

- Projekt muss bereits veröffentlicht sein (siehe Use Case *Projekt veröffentlichen*)

Nachbedingungen (bei Erfolg):

- neue Teilnehmer nehmen an der Programmiersitzung teil und werden im View *Shared Project Session* angezeigt
- neue Teilnehmer haben zunächst die Rolle Observer

Nachbedingungen (bei Misserfolg):

- Falls es bei der Synchronisation Probleme gab, dann muss der Originalstand (vor der Einladung) hergestellt werden

Standardablauf:

1. Host öffnet den Einladungsdialog (Button im View *Shared Project Session)*
2. Es wird eine Liste von verfügbaren Teilnehmern angezeigt
3. Host wählt den einzuladenden Teilnehmer aus und klickt auf den Button *Invite*
4. dem einzuladenden Teilnehmer wird eine Einladung geschickt und in der Liste wird der Status (warte auf Bestätigung) angezeigt
5. der eingeladene Teilnehmer akzeptiert die Einladung

6. Eine Dateiliste aller Dateien im Projekt werden zum eingeladenen Teilnehmer verschickt (Status in der Liste wird aktualisiert)
7. der eingeladene Teilnehmer wählt aus, ob in seinem Workspace ein neues Projekt angelegt werden oder ob ein bestehendes Projekt als Ausgangsbasis genommen werden soll
8. Es findet eine Synchronisation der Dateien statt, so dass auf beiden Seiten der gleiche Datenbestand vorliegt (Status in der Liste wird aktualisiert)
9. Nach erfolgreicher Synchronisation ist der Einladungsvorgang abgeschlossen und in der Liste wird als Status der Erfolg der Einladung angezeigt
10. Der Host wählt einen weiteren Teilnehmer aus (*Springe zu Schritt 3*) oder schließt den Einladungsdialog

Erweiterungen / alternative Abläufe:

- Alternativer Anfang 1: Gerade zuvor wurde ein Projekt veröffentlicht (Use Case 1) und der Einladungsdialog ist dadurch bereits geöffnet (Schritt 1 entfällt)
- Alternativer Anfang 2: Im Kontextmenu des Teilnehmers (View *Roster*) wurde „Invite user to shared project“ ausgewählt (Schritte 1-3 entfallen)
- 7a: Der eingeladene Teilnehmer wählt ein bestehendes Projekt als Ausgangsbasis und lässt für die verteilte Programmiersitzung eine Kopie des Projekts erstellen
- 7b: Der eingeladene Teilnehmer lässt vom Werkzeug ein bestehendes Projekt finden, dass vom Datenbestand am ähnlichsten ist. Der Grad der Übereinstimmung wird angezeigt. Danach hat er alle Möglichkeiten wir in *7/7a* beschrieben.
- 5a: Der eingeladene Teilnehmer lehnt die Einladung ab, dies wird als Status in der Liste angezeigt und zu Schritt 10 gesprungen
- 5b: Der Einzuladende ist bereits in einer anderen Programmiersitzung. Er wird über die Einladung informiert und kann entscheiden ob er die Sitzung wechseln möchte

spezielle Anforderungen:

- Für eine schnelle Synchronisation ist eine Peer-to-Peer Verbindung zwischen den Teilnehmern nötig

Häufigkeit des Auftretens: mittel

Use Case 3

Name: **Teilnehmer verlässt Sitzung**

Primärakteur: Teilnehmer

Stakeholder und Interessen:

- Teilnehmer: Will Programmiersitzung verlassen

Vorbedingungen:

- Teilnehmer nimmt an Sitzung teil (wird im View *Shared Project Session* angezeigt)

Nachbedingungen:

- Teilnehmer nimmt an Sitzung nicht mehr teil (wird bei allen Teilnehmern im View *Shared Project Session* nicht mehr angezeigt)

Standardablauf:

1. Teilnehmer drückt den Button zum Verlassen der Sitzung im View *Shared Project Session*

Erweiterungen / alternative Abläufe: siehe Use Case 4 (*Host verlässt Sitzung*)

spezielle Anforderungen: keine

Häufigkeit des Auftretens: mittel

Offene Fragen: keine

Use Case 4

Name: **Host verlässt Sitzung**

Primärakteur: Host

Stakeholder und Interessen:

- Host: Will Programmiersitzung verlassen

Vorbedingungen:

- Die Programmiersitzung wurde vom Host initiert

Nachbedingungen:

- Die Programmiersitzung wurde beendet, alle Teilnehmer habe Sitzung verlassen
- Datenbestand bei allen Teilnehmern sind konsistent

Standardablauf:

1. Host drückt den Button zum Verlassen der Sitzung im View *Shared Project Session*
2. alle Teilnehmer werden darüber informiert, dass Host Sitzung verlassen will und aufgefordert ihre Aktivitäten einzustellen
3. alle Teilnehmer bestätigen das Beenden der Programmiersitzung und verlassen damit die Sitzung
4. Datenbestand wird in einen konsistenten Zustand gebracht und die Programmiersitzung beendet
5. Erfolgsmeldung beim Host über das erfolgreiche Beenden der Sitzung
6. Alle Teilnehmer könne am Projekt lokal alleine weiter arbeiten

Erweiterungen / alternative Abläufe:

- 3a: Obwohl einige Teilnehmer das Beenden der Sitzung nicht bestätigten, erzwingt der Host den Prozess des Beendens
- alternativer Ablauf (Host ist übers Netz nicht mehr erreichbar)

1. Ist der Host für eine bestimmte Zeit nicht Erreichbar wird eine Warnmeldung bei allen Teilnehmern angezeigt
2. Die Funktionen für die Interaktion (Rollenwechsel, Nachrichten) werden deaktiviert
3. Teilnehmer können weiterarbeiten und die Veränderungen werden zwischengespeichert
4. Nachdem wieder eine Verbindung zum Host besteht werden die Teilnehmer darüber informiert, die zwischengespeicherten Änderungen werden abgearbeitet und die Funktionen für die Interaktion wieder aktiviert

spezielle Anforderungen: keine

Häufigkeit des Auftretens: mittel

Use Case 5

Name: **Textänderungen durchführen (einfache Editieroperationen)**

Primärakteur: Driver

Stakeholder und Interessen:

- Driver: Will gemäß seiner Rolle Änderungen am Quelltext vornehmen
- Observer / andere Driver: Wollen gemäß ihrer Rolle Änderungen am Quelltext mitbekommen

Vorbedingungen:

- es muss eine verteilte Programmiersitzung existieren, die gewünschten Teilnehmer bereits daran teilnehmen
- der Teilnehmer der Änderungen durchführen will muss Rolle Driver besitzen

Nachbedingungen:

- Änderungen müssen bei allen Teilnehmer vollzogen sein

Standardablauf:

1. Driver modifiziert den Quelltext (Lösch- oder Einfügeoperation)
2. Veränderung werden bei allen anderen Teilnehmer nachvollzogen (evtl. mit Transformation der Operation)

3. Awareness-Informationen (Veränderungen hervorheben, evtl. Sichtbereichsanzeige ändern , usw.) wird bei allen anderen Teilnehmern dargestellt

Erweiterungen / alternative Abläufe:

- Falls sich ein Observer im Verfolgermodus für diesen Driver befindet wird evtl. Fensterbereich gescrollt

spezielle Anforderungen:

- Die Nebenläufigkeitskontrolle muss die Konsistenz (Konvergenz, Kausalitäts- und Intentionserhaltung) sicherstellen

Häufigkeit des Auftretens: sehr häufig

Use Case 6

Name: **Verfolgermodus**

Primärakteur: Teilnehmer

Stakeholder und Interessen:

- beobachtender Teilnehmer: will einen anderen Teilnehmer (Driver oder Observer) verfolgen
- der zu beobachtende Teilnehmer: will im Rahmen einer kollaborativen Programmiersitzung beobachtet werden

Vorbedingungen:

- es muss eine verteilte Programmiersitzung existieren, die gewünschten Teilnehmer bereits daran teilnehmen

Nachbedingungen:

- der Fensterbereich des Editors beim Beobachter wird kontinuierlich so angepasst (durch evtl. scrollen) dass die Veränderungen am Quelltext bzw. das Selektieren von Textstellen beobachtet werden können
- Neben dem Namen des beobachtenden Teilnehmer wird im View *Roster* angezeigt, dass dieser verfolgt wird

Standardablauf:

1. beobachtender Teilnehmer: Rechtsklick im View *Shared Project Session* auf den zu beobachtenden Teilnehmer (Kontextmenu) und Auswählen der Aktion *follow*

Erweiterung (der beobachtete Teilnehmer verlässt Sitzung):

- Im View *Shared Project Session* wird signalisiert, dass der beobachtete Teilnehmer die Sitzung verlassen hat

Alternativen: keine

spezielle Anforderungen: keine

Häufigkeit des Auftretens: häufig

Use Case 7

Name: **Rollenwechsel**

Primärakteur: Host

Stakeholder und Interessen:

- Host: ist der einzige der die Rollenzuweisungen in der Programmiersitzung ändern darf
- Teilnehemer: möchte, dass seine Rolle gewechselt wird

Vorbedingungen:

- es muss eine verteilte Programmiersitzung existieren an der der Teilnehmer teilnimmt.

Nachbedingungen:

- der Rollenwechsel für den Teilnehmer wurde durchgeführt und wird im View *Shared Project Session* entsprechend angezeigt
- die Schreibberichtigung wird entsprechend der Rolle gesetzt

Standardablauf:

1. Host wählt im View *Shared Project Session* den Teilnehmer aus, deren Rolle gewechselt werden soll
2. im Kontextmenu wählt der Host eine der Aktionen zum geben oder zum wegnehmen der Driver Rolle aus

Erweiterungen / alternative Abläufe: keine

spezielle Anforderungen: keine

Häufigkeit des Auftretens: häufig

Offene Fragen: keine

Use Case 8

Name: **Kommunikation**

Primärakteur: Teilnehmer

Stakeholder und Interessen:

- Teilnehmer: möchte mit anderen Teilnehmer kommunizieren

Vorbedingungen:

- es muss eine verteilte Programmiersitzung existieren, die gewünschten Teilnehmer bereits daran teilnehmen

Nachbedingungen:

- die Nachrichtist an alle anderen Teilnehmer verschickt worden
- die Nachricht wurden bei allen anderen Teilnehmern angezeigt

Standardablauf

1. Teilnehmer: öffnet den View *Chat (*falls dieser nicht geöffnet ist)
2. Ein View zum Empfangen und Versenden von Text-Nachrichten wird angezeigt
3. im unteren Teil des Views wird die zu versendende Nachricht eingeben
4. Bei allen anderen Teilnehmern wird (falls noch nicht bereits geöffnet) der View geöffnet und die Nachricht im oberen Teil angezeigt (mit Nickname des Senders und dessen Hintergrundfarbe)
5. Der Empfänger kann auf die Nachricht antworten (gehe zu Schritt 3) oder nicht antworten und den View entweder offen halten oder schließen

Schritte 1-2 können entfallen, falls der View bereits geöffnet

Alternativer Ablauf (Skype Kommunikation):

1. Teilnehmer: Rechtsklick im View *Shared Project* Session auf den gewünschten Kommunikationspartner *(Kontextmenu)* und Auswählen der Aktion *Skype this User*
2. zu dem ausgewählten Teilnehmer wird eine Verbindung mit Skype aufgebaut (es wird das Programm Skype aufgerufen und ein Gruppenunterhaltung mit dem ausgewählten Teilnehmer geöffnet)

spezielle Anforderungen: keine

Häufigkeit des Auftretens: häufig

Use Case 9

Name: **Durchführung von Dateioperationen oder komplexen Editieroperationen (Refactorings)**

Primärakteur: Driver

Stakeholder und Interessen:

- Driver: Will eine Dateioperation oder komplexere Editieroperation (Refactoring) durchführen

Vorbedingungen:

- es muss eine verteilte Programmiersitzung existieren, die gewünschten Teilnehmer bereits daran teilnehmen
- der Teilnehmer der Änderungen durchführen will muss Rolle Driver besitzen

Nachbedingungen:

- bei allen Teilnehmer müssen die Veränderungen vollzogen worden sein

Standardablauf:

1. Driver wählt wie in Eclipse gewohnt Dateioperationen oder Refactoring aus
2. Falls kein andere Driver betroffene Datei geöffnet hat, springe zu Schritt 5
3. Alle anderen Driver die eine betroffene Datei geöffnet haben werden darüber informiert, dass eine Dateioperation oder ein Refactoring durchgeführt werden soll, das ein exklusives Schreibrecht erfordert
4. Alle anderen Driver geben ihre Zustimmung
5. für alle anderen Driver werden betroffene Dateien gesperrt (d.h. der Schreibzugriff wird entzogen)
6. Veränderungen werden bei allen Teilnehmern durchgeführt
7. Während der Durchführung werden die Dateien in der Dateiliste als gesperrt markiert
8. Nach der Durchführung bekommen alle Driver wieder für die betroffenen Dateien Schreibzugriff
9. Markierungen in der Dateiliste werden entfernt

Erweiterung (während der Durchführung will jemand eine betroffene Datei öffnen)::

- derjenige wird darüber informiert, dass gerade eine Sperre auf der Datei ist und er diese nur lesend öffnen kann

alternative Abläufe: keine

spezielle Anforderungen: keine

Häufigkeit des Auftretens: mittel