Exposé zur Masterarbeit

Verbesserung der Action-Awareness in Saros

Damla Durmaz

4. Mai 2014

1 Einleitung

Bei der klassischen Paarprogrammierung sitzen beide Programmierer nebeneinander an einem Computer, der eine Teilnehmer programmiert (**Driver**), während der andere Teilnehmer zuschaut, sich Gedanken über das Problem macht, den Quellcode kontrolliert und über Fehler spricht, falls diesem welche auffallen (**Observer**) [1]. Diese Rollenverteilung ist jedoch nicht fest. Nach Stotts et al. in [1] macht eine effektive Paarprogrammierung aus, dass diese Rollen in periodischen Zeitabständen wechseln. Während der Observer die Tätigkeiten des Drivers beobachtet und kommentiert, hängen seine Eindrücke, Gedanken und Aktionen von denen des Drivers ab [2]:

- Wo im Quellcode befindet sich die Person?
- Was macht diese Person an der entsprechenden Stelle?
- Was interessiert die Person gerade?
- Was sind die Absichten?

Die Aktionen des Beobachters sind abhängig von diesen Informationen. Dieses Wissen über die Aktionen und den Status anderer Personen in der Gruppenarbeit wird auch als „Awareness“ bezeichnet und bildet die Grundlage für die eigenen Aktionen [3]. Es handelt sich dabei *um auf die Minute aktuelles Wissen, um die Aktivitäten anderer Menschen, das vom einzelnen benötigt wird, um sich mit diesen zu koordinieren und die eigenen Aufgaben*

innerhalb der Gruppe zu erfüllen und kann die Qualität der kollaborativen Zusammenarbeit beeinflussen [4].

Wie wichtig Awareness Informationen sind, soll folgendes Beispiel verdeutlichen: Zwei Studenten möchten gemeinsam ein Plakat für eine Präsentation vorbereiten. Dabei teilen sie die Arbeit auf, wobei jeder von ihnen eine Hälfte des Plakats bearbeitet. Während jeder Student an seiner Hälfte arbeitet, schaut jeder von ihnen gelegentlich über die Schulter zum anderen Studenten und kann dabei folgende Dinge in Erfahrung bringen:

- Was macht der andere gerade?
- Wo auf dem Plakat ist er gerade? Wo darf der beobachtende Student also nicht schreiben?
- Was könnte er gleich machen? Braucht er gleich eine Schere oder den Stift, den der Beobachtende gerade nutzt?
- Hat der beobachtende Student Anmerkungen zu der Arbeit des anderen?

Wenn ein Student weiß, wo auf dem Plakat sich der andere befindet, kann er z.B. nachfragen, wie der andere seinen Teil gestalten möchte oder wie weit dieser mit seiner Arbeit ist. Evtl. hat der Student eine Idee, wie man Elemente anordnen könnte. Würden die Studenten nichts von dem wissen, würden sie nicht wissen, wie weit der andere ist, sie könnten kein Feedback geben oder sie würden sogar Sachen machen, die nicht vorher besprochen wurden. Es könnte sogar so weit kommen, dass sie am Ende ihrer Arbeit die beiden Hälften des Plakats zusammenlegen und feststellen, dass diese gar nicht zusammenpassen. Würden die Studenten Awareness-Informationen erhalten, die jedoch schwer zu interpretieren sind, würde das auch wenig weiterhelfen. Wenn sie sich z.B. gegenseitig nicht sehen, jedoch erfahren, wenn der andere mit einer neuen Spalte beginnen, können sie daraus nicht schlussfolgern, was sie jetzt schreiben werden, wo die neue Spalte auf dem Plakat angeordnet wird usw. Awareness ist also nicht nur wichtig, um in einer Gruppe koordiniert zu arbeiten und die Situation zu erfassen, sondern kann deutlich die Qualität dieser Koordination beeinflussen.

Bei der kollaborativen Programmierung ist eine spezielle Form von Awareness von Bedeutung, und zwar „Workspace Awareness“. Das gemeinsam genutzte Medium ist der Arbeitsbereich und die Awareness-Informationen sind hier die Informationen über die Anwesenheit, die Aktivitäten und die Absichten der Nutzer in diesem gemeinsamen Arbeitsbereich. Ist die kollaborative Programmierung verteilt, so sind Workspace Awareness Mechanismen nicht wegzudenken, da hier die Teilnehmer der Kollaboration nicht

am selben Standort sind und somit nur im gemeinsamen Arbeitsbereich die Interaktionen der anderen Nutzer mit diesem erfassen können. Die Kollaboration wird demnach problematischer und kann qualitativ beeinträchtigt werden [5]. Wenn bei dieser verteilten Kollaboration dazu mehr als zwei Teilnehmer vorhanden sind, wechseln die Teilnehmer während der Kollaboration immer wieder ihre Aufmerksamkeit zwischen der gemeinsamen Arbeit und der individuellen, unabhängigen Arbeit. Wenn keine Workspace Awareness Mechanismen zur Verfügung stehen würden, würden die Teilnehmer nicht mehr wissen, woran die anderen arbeiten, worauf sie sich konzentrieren und was ihre Absichten im Arbeitsbereich sind [6]. Werden jedoch ausreichend Workspace Awareness Informationen zur Verfügung gestellt, kann die verteilte, kollaborative Programmierung unterstützt werden und zu einer qualitätsvolleren Koordination führen.

2 Problemstellung

Saros ist ein für die Entwicklungsumgebung Eclipse entwickeltes Open-Source-Plugin für die verteilte, kollaborative Programmierung [7] und überträgt Awareness Informationen aus dem Code-Editor (wer tippt gerade was und wo). Diese Art von Workspace Awareness wird auch als „Action Awareness“ bezeichnet [8] und gibt Informationen über die Aktivitäten und Absichten der anderen Teammitglieder wieder. Action Awareness deckt Fragen ab, wie „Was passiert?“, „Welche Aktionen werden von wem ausgeführt?“ und auch „Was ist bisher passiert“ [9]. Saros überträgt jedoch keine Informationen über die Aktivitäten, die in anderen Anwendungen, anderen Eclipse Views als dem Editor oder Dialogen (Suche, Refactoring etc.) stattfinden.

Action Awareness kann auf unterschiedlichen Ebenen bestehen, entweder ganz allgemein „Der andere ist in einer anderen Anwendung“ oder sehr detailliert „Der andere macht gerade ein Refactoring in Datei X in Zeile Y“. Fehlen diese Informationen, kann in der kollaborativen Arbeit nicht mehr nachvollzogen werden, was gerade von wem getan wird und was bisher bereits getan wurde. Wenn z.B. ein Programmierer bei der verteilten Paarprogrammierung nicht weiß, dass der Partner gerade bereits den Namen der Klasse refactort und dies zur selben Zeit ebenfalls durchführt, kann es zu Konflikten kommen.

3 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Erarbeitung eines Konzepts zur Verbesserung der Action Awareness Mechanismen in Saros. Als Grundlage für die Erar-

beitung des Konzepts werden Videos analysiert, in denen mehrere Saros-Sitzungen zwischen zwei Programmierern aufgezeichnet wurden, die an unterschiedlichen Standorten arbeiten. Durch die Analyse soll der Bedarf für Action Awareness Mechanismen in Saros ermittelt werden, sodass aus den gefundenen Bedarfsmöglichkeiten Anforderungen herausgezogen werden und somit ein Konzept entwickelt werden kann. Dieses Konzept wird anschließend implementiert und in einer Feedback-Runde evaluiert. Die Evaluation dient dazu, das implementierte Konzept mit realen Nutzern zu testen, um zu erkennen, ob diese von den neuen Action Awareness Mechanismen profitieren.

4 Abgrenzung der Arbeit

In der Literatur sind einige Mechanismen verbreitet, die Action Awareness unterstützen. Gutwin und Greenberg haben in [8] und Schümmer und Lukosch in [10] einige vorgestellt:

- **Indikator für Aktivitäten und Änderungen**: Hier werden auf der Benutzeroberfläche Symbole als Indikatoren für Aktivitäten oder Änderungen genutzt. Diese Indikatoren können auch Zahlenwerte sein, um die Menge an Änderungen darzustellen.

- **Telepointer**: In der natürlichen Kommunikation wird der Mauszeiger verwendet, um als Ersatz für den Finger zu dienen. Oft wird der Zeiger auch für deiktische Ausdrücke („das da“, „hier“, „dort“) verwendet und sagt aus, wo sich der Akteur befindet und auch was er tut.

- **Entfernter Mauszeiger**: Der Telepointer ist ein visueller Pointer und unterscheidet sich von einem entfernten Mauszeiger darin, dass der Telepointer unabhängig ist von der Eingabe- und Interaktionsfähigkeit. Mit dem entfernten Mauszeiger kann man also Artefakte im entfernten Arbeitsbereich verschieben und verändern, während der Telepointer lediglich dazu dient, auf einen Punkt im Arbeitsbereich zu zeigen.

- **Sichtbarkeit der Aktivitäten**: Hier werden die Aktivitäten der anderen Teilnehmer sichtbar gemacht, indem z.B. um den Aufenthaltsort herum ein kleines Sichtfeld angezeigt wird, sodass man sehen kann, wo diese Person im Arbeitsbereich ist. Daraus kann schlussgefolgert werden, was diese Person gerade tut.

- **Radar Views**: Eine andere Möglichkeit, um Aktivitäten sichtbar zu machen, sind die sogenannten Radar Views, bei denen der gesamte Arbeitsbereich der anderen in einer kleinen Miniatur angezeigt wird,

die wiederum selbst Awareness Informationen enthalten kann. Diese können Wissen über die Artefakte, die die anderen nutzen, geben, aber auch die Aktionen mit diesen Artefakten und die Absichten, die mit der Arbeit an diesen Artefakten verfolgt werden.

Die in dieser Arbeit zu entwickelnden Mechanismen grenzen sich von oben beispielhaft aufgezählten Mechanismen in der Art ab, dass sie die Action Awareness rein textuell übertragen. Die obigen Mechanismen sind interessante und teilweise bewährte Ansätze und können im Ausblick der Arbeit näher betrachtet werden, werden aber nicht in dieser Arbeit umgesetzt. Diese Arbeit legt den Fokus auf die textuelle Übertragung von Action Awareness Mechanismen, weil zum einen kein Mechanismus implementieren werden soll, der durch zusätzliche Views nur stören würde und zum anderen sind bisher keine der bekannten oder andere Lösungen aus der Literatur bekannt, die solche feingranularen Informationen übertragen. Die Form, in der die Texte bzw. die Action Awareness Informationen in Saros präsentiert werden, steht noch offen und wird im Laufe der Arbeit definiert.

5 Durchzuführende Schritte/Aufgabe

- Literaturrecherche
  - Groupawareness
  - Workspace Awareness
  - Action Awareness
  - Grounded Theory
- Einarbeitung in die theoretischen Grundlagen
- Erarbeitung des Grundwissens für die Analyse der Videos (Grounded Theory)
- Analyse der Videos
  - Ansehen der Videos
  - Gefühl dafür entwickeln, welche Stellen für mich wichtig sind
  - Bedarf für Action Awareness Mechanismen erkennen
- Erarbeitung eines Konzepts
  - An welchen Stellen liegt Bedarf vor?

  - Was wäre dort wichtig, zu übertragen?
  - Was möchte ich übertragen?
  - Wie kann man das begründen?
  - (Grob) Was kann man technisch umsetzen?
- Einarbeitung in die wichtigen Stellen vom Saros Code
  - Welche Informationen kann ich in Eclipse abgreifen?
  - Ist mein Konzept technisch realisierbar?
- Anpassung des Konzepts
  - Überarbeitung des Konzepts anhand des Wissens der technischen Realisierbarkeit
  - Überlegung eines Mechanismus zur Darstellung der Action Awareness Mechanismen
- Implementierung des Konzepts
- Evaluation des Konzepts (wird evtl. in mehreren Iterationen stattfinden)
  - Überlegung eines Konzepts für Nutzertests
  - Durchführung der Nutzertests
  - Analyse der Nutzertests
  - Einarbeitung des Feedbacks oder Teile davon
- Überlegung weiterführender Arbeiten

6 Struktur der Arbeit

Das Inhaltsverzeichnis der Arbeit ist in der folgenden Form nicht garantiert, da die Anzahl an nötigen Nutzertests zur Erfassung der Anforderungen an die Implementierung nicht bekannt ist. In diesem Google Docs Dokument befindet sich das aktuelle Inhaltsverzeichnis und wird im Laufe der Arbeit stets aktualisiert, wenn Änderungen nötig werden.

7 Mitarbeit im Saros-Projekt

- Mitarbeit am Open-Source-Projekt Saros
  - Entwicklung von Action Awareness Mechanismen
  - Dokumentation der implementierten Mechanismen im Code
- Mitarbeit am Release-Prozess von Saros
  - Code-Reviews

Literatur

[1] David Stotts, Jason McC. Smith, Prashant Baheti, Prashant Baheti, Laurie Williams, Laurie Williams, Edward Gehringer, and Edward Gehringer. Distributed pair programming: Empirical studies and supporting environments. Technical report, 2002.

[2] J. Schenk. Evaluating awareness information in distributed collaborative editing by software-engineers. In *User Evaluation for Software Engineering Researchers (USER), 2012*, pages 35–38, June 2012.

[3] Carl Gutwin, Carl Gutwin, Saul Greenberg, and Saul Greenberg. Support for group awareness in real-time desktop conferences. In *University of Waikato*, pages 18–21, 1995.

[4] Carl Gutwin, Saul Greenberg, and Mark Roseman. Supporting awareness of others in groupware. In *Conference Companion on Human Factors in Computing Systems*, CHI '96, pages 205–, New York, NY, USA, 1996. ACM.

[5] Carl Gutwin, Saul Greenberg, and Mark Roseman. Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. In *Proceedings of HCI on People and Computers XI*, HCI '96, pages 281–298, London, UK, UK, 1996. Springer-Verlag.

[6] Carl Gutwin and Saul Greenberg. Workspace awareness for groupware. In *Conference Companion on Human Factors in Computing Systems*, CHI '96, pages 208–209, New York, NY, USA, 1996. ACM.

[7] Stephan Salinger, Christopher Oezbek, Karl Beecher, and Julia Schenk. Saros: An eclipse plug-in for distributed party programming. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of*

Software Engineering, CHASE '10, pages 48–55, New York, NY, USA, 2010. ACM.

[8] Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Comput. Supported Coop. Work*, 11(3):411–446, November 2002.

[9] Lu Xiao. *The Effects of Rationale Awareness in Hybrid Collaborative Learning Activities*. PhD thesis, University Park, PA, USA, 2008. AAI3336152.

[10] T. Schummer and S. Lukosch. *Patterns for Computer-Mediated Interaction*. Wiley Software Patterns Series. Wiley, 2007.