Freie Universität Berlin

Unterstützung unterschiedlicher Programmier-sprachen und Plugins in einem Werkzeug für kollaborative Softwareentwicklung

Bachelorarbeit von Alena Kiwitt

- Themenüberblick und Motivation
- Ziele
- Anforderungsanalyse
- Testfälle und Durchführung der Tests
- Ergebnis der Kompatibilitätstests
- Fazit
- Ausblick

- Eclipse IDE & Saros Plugin
  - Eclipse: flexible Entwicklungsumgebung, durch Plugins individuell anpassbar und erweiterbar, weit verbreitet
  - Saros: Eclipse-Plugin für kollaborative Softwareentwicklung, speziell Paarprogrammierung
- Unterstützung verschiedener Programmiersprachen
  - Saros wird unter Eclipse Classic entwickelt, unterstützt also standardmäßig zunächst Java
  - Unterstützung anderer Programmiersprachen wäre wünschenswert
    - Möglichkeiten von Eclipse ausnutzen (zahlreiche Sprach-Plugins)
    - Viele Programmierer erreichen, Verbreitung von Saros steigern
    - Feedback der Programmierer zur Optimierung von Saros einsetzen

Freie Universität Berlin

- Auswahl verschiedener Plugins, basierend auf der bestehenden Kompatibilitätsliste
- Anforderungsanalyse sowohl der Plugins als auch von Saros hinsichtlich der Kompatibilität
- Entwurf von Testfällen und eines Testszenarios, um die Vergleichbarkeit der Tests zu gewährleisten
- Testen der ausgewählten Plugins zur Aktualisierung und Ergänzung der bestehenden Kompatibilitätsliste

- Bestehende Kompatibilitätsliste
- Eclipse Marketplace bietet Informationen zu
  - Aktualität
  - Aktivität (sowohl von Anwendern als auch von Entwicklern)
  - Beliebtheit
- Tiobe-Index:
  - Liste der beliebtesten Programmiersprachen
- Kombination aus beliebtesten Sprachen & Plugins
  - Für einige Sprachen sind keine Plugins verfügbar
  - Auch berücksichtigt: ist ein Plugin kommerziell oder frei verfügbar?

- Ermittlung von Kompatibilitätskriterien, die ein Plugin erfüllen muss
- Betrachtung von beiden Seiten
  - Aus Sicht von Saros
  - Aus Sicht des jeweils ausgewählten Plugins
- Sowohl Saros als auch das Plugin stellen bestimmte Funktionalität zur Verfügung
  - Welche Saros-Features müssen unbedingt erfüllt werden?
  - Welche Plugin-Features berühren die Verwendung von Saros?
- Außerdem: wie ermittelt man relevante Funktionalitäten eines Plugins, mit dem man noch nie gearbeitet hat?

- Unterteilung der Funktionalitäten in folgende Bereiche:
  - Installation und Aufbau einer Saros-Sitzung
  - Benutzerrollen
  - Awareness-Informationen
  - Der Follow Mode
  - Dateioperationen und Refactoring
  - Vermeidung und Auflösung von Inkonsistenzen
- Zu jeder Funktionalität wurden konkrete Anforderungen formuliert und deren Gewichtung festgehalten
  - z.B. sind Anforderungen immer dann als besonders kritisch / wichtig einzustufen, wenn die Modifikation und somit Synchronisation von Quelltext betroffen ist

- Annahme zunächst: es lässt sich eine allgemein gültige Methode finden, um Funktionalitäten und damit auch mögliche Anforderungen der verschiedene Plugins zu identifizieren
  - Dem ist leider nicht so...
- Aber: Ein Großteil der Funktionalitäten überschneidet sich
- Funktionalitäten wurden daher zu Feature-Klassen zusammengefasst:
  - Ausführung des Programmcodes
  - Werkzeuge zur Quelltext-Analyse
  - Werkzeuge zur automatischen Quelltext-Modifikation
- Gelten für alle Sprach-Plugins
  - Außerdem: Hinweise zur Bewertung und Erleichterung der Einordnung verschiedener Plugin-Features in die Feature-Klassen

Freie Universität Berlin

- Konkrete Testfälle zur Überprüfung der Kompatibilität aus Sicht von Saros
  - Einordnung entsprechend der in der Anforderungsanalyse ermittelten Bereiche
- Generische Testfälle zur Überprüfung der Kompatibilität aus Sicht der verschiedenen Plugins
  - Orientierung anhand der Feature-Klassen
- Beschreibung eines Testszenarios
  - Hält man sich beim Testen daran, werden die Tests vergleichbar
  - Der Begriff der Kompatibilität wird genau(er) gefasst
  - Einflüsse durch vom Test unabhängige Einstellungen werden vermieden

- Insgesamt erfreulich:
  - Die Mehrheit der getesteten Plugins ist kompatibel mit Saros
  - Bei Inkompatibilität liegt die Ursache meist nicht bei Saros
- Wenn Tests fehlschlugen, dann meist schon ganz zu Beginn
  - Gleichzeitige Installation nicht möglich / Versionskonflikte
  - Hier lag die Ursache ebenfalls durchweg nicht bei Saros
- Kleinere Inkonsistenzen traten bei den Awareness-Informationen auf
  - Diese sind als eher unkritisch zu bewerten, da die Quelltext-Synchronisation weiterhin gegeben ist
- Erkenntnis: zur besseren Unterstützung von HTML-Editoren u.ä. wäre eine konsequentere Durchsetzung von Schreibsperren wünschenswert

Freie Universität Berlin

- Anforderungsanalyse für Saros-Features verlief recht geradlinig
  - Genaue Kenntnis über Features, Funktionsweise, Relevanz, etc.
- Der „Rundumschlag“ für die Sprach-Plugins ist leider nicht gelungen
  - Resonanz auf Anfragen an die Communitys ließ sehr zu wünschen übrig
  - Positiv: für viele Plugins existieren detaillierte Feature-Listen
- Durch die Feature-Klassen wird der wesentliche Teil der Plugin-Features abgedeckt
  - Lediglich mögliche Spezial-Features müssen weiterhin individuell betrachtet werden (ein Vergleich ist hier aber auch nicht sinnvoll)
- Insgesamt: bessere Transparenz, konkrete Anforderungen & Testfälle, auf die zurückgegriffen werden kann, aktualisierte Kompatibilitätsliste

- Damit die Kompatibilitätsliste aussagekräftig bleibt, muss sie aktuell gehalten werden
  - Ändern sich Features, müssen Testfälle evtl. angepasst werden
  - Saros und auch die getesteten Plugins werden stetig weiterentwickelt
- Der Anforderungskatalog muss gepflegt werden
  - Neue Saros-Features & Plugin-Features müssen aufgenommen werden
- Testfälle müssen bei neuen Anforderungen ergänzt werden
- Wünschenswert wäre zukünftig eine automatisierte Durchführung der Testfälle, siehe Sandors Test-Framework
- Das Wissen über die Kompatibilität von Saros zu anderen Plugins sollte genutzt werden:
  - Saros könnte gezielt in anderen Communitys bekannt(er) gemacht werden

DANKE!