Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. B. Beckert
Michael Kirsten, Mihai Herda

Praxis der Softwareentwicklung, WS 2017/18

Aufgabenbeschreibung

Eine kurze Bemerkung vorab

Dies ist *Ihr* Projekt. Dieses Dokument ist kein Katalog von Aufgaben, der Punkt für Punkt abgearbeitet werden muss, um das Modul zu bestehen, sondern lediglich eine Reihe von Hinweisen, was wir erwarten. Wie *Ihr* Programm nachher aussieht, müssen Sie selbst entscheiden.

1 Hintergrund

Eines der Hauptforschungsthemen an unserem Lehrstuhl ist die Verifikation relationaler Eigenschaften von Programmen. Relationale Eigenschaften sind solche Eigenschaften, die sich auf mindestens zwei Programmausführungen beziehen. Ein Beispiel dafür ist die Programmäquivalenz: Liefern zwei Programme bei gleicher Eingabe immer die gleiche Ausgabe? Ein anderes wichtiges Beispiel für relationale Eigenschaften stellen Informationsflusseigenschaften dar: Haben gewisse Eingaben Einfluss auf gewisse Ausgaben? Eine besondere Rolle spielt hier die Nichtinterferenz-Eigenschaft (engl. noninterference). Diese besagt, dass als geheim eingestufte Eingaben (z. B. ein Passwort) keine öffentlichen Ausgaben beeinflussen können. Für den Nachweis solcher Informationsflusseigenschaften wird eine Technik namens Selbstkomposition verwendet, wobei zwei Ausführungen desselben Programms mit verschiedenen geheimen Eingaben miteinander verglichen werden.

Unser Lehrstuhl hat bereits verschiedene Ansätze entwickelt und Tools (z. B. llrêve, llrêve for noninterference oder KeY) implementiert, um relationale Eigenschaften formal beweisen zu können. Da es sich hierbei allerdings um ein unentscheidbares Problem handelt und folglich ein Beweis nicht immer gelingen kann, möchten wir im Rahmen dieses PSE-Projekts ein Werkzeug entwickeln, das dem Benutzer die Möglichkeit gibt, mehrere Programausführungen interaktiv zu analysieren und zu vergleichen. Hierzu soll ein sogenannter relationaler Debugger entwickelt werden, der im Unterschied zu einem herkömmlichen Debugger mehrere Programmläufe nebeneinanderlegen und vergleichen kann.

2 Aufgabenstellung

Entwickeln Sie ein Software-System entsprechend den oben genannten Zielen. Im Folgenden sind einige Eckpunkte beschrieben.

2.1 Minimale Leistungsmerkmale

Debugging. Der relationale Debugger soll die gleichzeitige Analyse von mindestens zwei Programmabläufe ermöglichen. Dabei sollen die üblichen Funktionalitäten eines Debuggers für jeden analysierten Programmlauf zur Verfügung stehen.

Dazu zählen:

- Break-points an beliebigen Stellen im Code,
- Stepping (step into, step over, step out),
- Anzeige des aktuellen Programmzustands.

Relationale Analyse. Der Vergleich von den analysierten Programmläufen soll durch folgende Funktionalitäten erleichtert werden:

- Gleichzeitiges Stepping in den untersuchten Programmläufen. Für Schleifen soll es dem Benutzer möglich sein, die Anzahl der Iterationen anzugeben.
- Der Benutzer soll die Programmteile, die er vergleichen möchte, markieren können. Einfache Tests für diese Programmteile sollen unterstützt werden und die Ergebnisse dieser Tests sollen, wenn verfügbar, angezeigt werden.
- Conditional break-points: Die zwei Programmläufe werden ausgeführt, bis eine vom Nutzer angegebene Bedingung erfüllt wird.

Grafische Oberfläche. Ihr Programm muss über eine grafische Benutzeroberfläche verfügen. Dabei sollen die analysierten Programme, deren Eingaben und deren jeweiligen aktuellen Programmzustand angezeigt werden.

Zielsprache. Die Zielsprache soll eine Teilmenge der Programmiersprache Java sein. Es sollen Methoden, primitive Typen, Arrays von primitiven Typen und Operationen dafür unterstützt werden.

Software-Architektur. Eine saubere Trennung zwischen kritischen (z.B. Debugging) und unkritischen (z.B. GUI) Komponenten ist von immenser Bedeutung. Entwerfen Sie austauschbare Komponenten mit minimalen, wohldefinierten Schnittstellen.

2.2 Technische Eckpunkte

- Programmiersprache: Java
- GUI-Bibliothek: Swing
- Versionsverwaltung: Subversion (siehe unten)
- Entwurfs- und Entwicklungsumgebung: nach Wunsch, z.B. ArgoUML
- Unit-Test-Rahmenwerk: JUnit
- Dokumenterstellung: nach Wunsch, bevorzugt LaTeX (Abgabe aber immer als PDF)

3 Organisatorisches

3.1 Technische Ausstattung

Die Website zur Veranstaltung findet sich unter `https://formal.iti.kit.edu/teaching/pse/201718`. Dort werden in Zukunft weitere organisatorische Information bereitgestellt.

Wir stellen ein SVN-Repository bereit, in dem alle Artefakte (insbesondere Abgaben) abgelegt werden sollen. Vor der Nutzung müssen Sie sich erst registrieren, dazu besuchen Sie folgende Website: `https://svnserver.informatik.kit.edu/i57/login/` und loggen sich mit Ihrem ATIS-Account (`s_...`) ein; der Account wird dann von uns freigeschaltet.

3.2 Bewertung

Die Benotung Ihres Systems richtet sich nach folgenden Kriterien:[1]

- Qualität aller abgegebenen Dokumente
- Qualität der Kolloquien
- Qualität der Abschlusspräsentation
- Erfüllen der minimalen Leistungsmerkmale (s.o.)
- *Sinnvolle*[2] Erweiterungen über diese Merkmale hinaus
- Qualität des erstellten Programms (das schließt u.A. Benutzbarkeit und Robustheit ein)

Die Gesamtnote errechnet sich dann nach der im Modulhandbuch genannten Gewichtung:

- Pflichtenheft 10%
- Entwurf 30%
- Implementierung 30%
- Qualitätssicherung 20%
- Abschlusspräsentation 10%

Nach jeder Phase findet (im Rahmen der regelmäßigen Treffen) ein Kolloquium statt, in dem die Ergebnisse der Phase *selbstständig* (in rund 20 Minuten) vorgestellt werden sollen. Jedes Team-Mitglied muss einmal präsentieren.[3] Die Benotung jeder einzelnen Phase wird im entsprechenden Kolloquium besprochen.

3.3 Treffen

Treffen finden wöchentlich im Raum 211 statt. Auch wenn gerade kein Kolloquium ansteht, raten wir Ihnen dringend, jede Woche über Ihren Fortschritt zu berichten. Nur dann können wir unverbindliche, gezielte Kommentare abgeben. Sorgen Sie daher bitte auch dafür, dass alle erforderlichen Dokumente rechtzeitig im SVN verfügbar sind. Die Abgabeversion soll eindeutig identifizierbar sein. Halten Sie sich an den vereinbarten Abgabeschluss. Eine Woche vor Abgabe ist eine Vorversion einzureichen.

Darüber hinaus sollten Sie sich natürlich auch noch regelmäßig in der Gruppe besprechen. Sie können dazu Raum 211 nutzen falls er frei ist (vor der Tür hängt ein Kalender mit Reservierungen; bitte nehmen Sie Rücksicht, dass MitarbeiterInnen Priorität genießen).

[1] Diese Liste hat keine Reihenfolge, die einer Gewichtung entspricht. Es gibt sicherlich weitere Punkte, die als selbstverständlich gelten und sich bei Nichterfüllen negativ auswirken.

[2] Sie tun Sich selbst und dem Projekt nichts Gutes wenn Sie sich zu sehr verkünsteln.

[3] Dennoch müssen sich selbstverständlich *alle* Team-Mitglieder in *allen* Phasen einbringen.