Software Engineering

Department of Computer Science
Software Technology Group
Dr. Michael Eichberg, Leonid Glanz, Sven Amann

Übungszettel 05: Generelle Informationen zur Abgabe

Abgabefrist: 03.12.2015, 08:00 Uhr
(Alle Lösungen müssen bis zu den jeweiligen Zeitpunkten eingereicht sein)

Erzeugen Sie aus den Lösungen aller Teilaufgaben, die keinen Quelltext erfordern, **eine** Datei „solution_xxx.pdf“. Fügen Sie für xxx Ihre Gruppennummer ein. Gliedern Sie die Abgabedatei den Teilaufgaben entsprechend. Achten Sie auf korrekte Rechtschreibung und Grammatik. Die Tutoren sind angewiesen chaotische, unverständliche oder fehlerüberhäufte Lösungen zu ignorieren.

Alle Lösungen sind in Moodle einzustellen. Abgaben per E-Mail oder auf anderem Weg werden nicht berücksichtigt. Beachten Sie unbedingt die in der Übung vorgestellten Namenskonventionen. Ihre Abgabe wird nicht bewertet, wenn der Tutor die Lösungen nicht finden oder nicht zuordnen kann.

Die Korrekturen werden ebenfalls in Moodle eingestellt. Wir versuchen dies im Normalfall eine Woche nach Abgabe der Übung zu schaffen. Wenn Sie Fragen bzgl. der Korrekturen haben, schreiben Sie dem korrigierenden Tutor eine E-Mail.

Software Qualitätswerkzeuge

Laden Sie sich den Source Code von Apache Commons aus Moodle und analysieren Sie ihn mit den in der Vorlesung aufgeführten Werkzeugen.

Laden Sie dich hierzu die Werkzeuge:

- FindBugs (http://findbugs.sourceforge.net/downloads.html oder Help->Eclipse Marketplace->Find: Findbugs)
- PMD (https://pmd.github.io/ oder Help->Eclipse Marketplace->Find: PMD oder Help-> Install New Software-> Workin with:
  http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/ -> Add)
- CheckStyle (http://checkstyle.sourceforge.net/ oder Help->Eclipse Marketplace->Find: checkstyle)
- JDepend (http://clarkware.com/software/JDepend.html oder Help->Eclipse Marketplace->Find: jdepend)
- Dependency Finder (http://depfind.sourceforge.net/)
- Checker Framework (http://types.cs.washington.edu/checker-framework/)

herunter.

Aufgabe 1: Werkzeug Verständnis (4 Punkte)

a) Beschreiben Sie in eigenen Worten welchen Zweck jedes der einzelnen Werkzeuge erfüllt.
Hinweis: Beschreiben Sie das Werkzeug kurz und präzise.
Beispiel: Bugpicker (http://www.opal-project.de/tools/bugpicker/) verwendet fortgeschrittene statische Analysen, um komplexe Daten- und Kontrollflussprobleme zu identifizieren.

b) Beschreiben Sie in welchen Fällen das Werkzeug unpräzise (false Negatives/ false Positives) werden kann.
Hinweis: Betrachten Sie dabei die Datenstrukturen auf denen das jeweilige Werkzeug arbeitet.
Beispiel: Der Bugpicker arbeitet auf Java Bytecode und extrahiert Daten- und Kontrollflussgraphen. Im Falle von komplexen Abhängigkeiten zwischen lokalen Variablen und/ oder Feldern kann es zu false Negatives oder false Positives kommen. Siehe Foliensatz Softwarequalität Folie 46.

Aufgabe 2: Werkzeug Ausführung (10 Punkte)

a) Führen Sie die Werkzeuge FindBugs, PMD, CheckStyle und JDepend auf dem Source Code aus, welcher in Moodle zu finden ist und von Apache Commons abgeleitet wurde. Beschreiben Sie je drei Probleme aus der höchst möglichen Kategorie, die durch das jeweilige Werkzeug gefunden werden.
Hinweis: Beschreiben Sie die genaue Stelle des auftretenden Problems und ob es ein Problem darstellt.
Beispiel: Findbugs: In der Klasse …/chain/web/ChainListener wird in der Zeile 353 eine mögliche Null-Pointer-Dereference gefunden. Das Problem ist, sollte in Zeile 323 eine Exception geworfen werden, dann ist die resourceURL tatsächlich null. Es handelt sich um einen **true Positive**.

b) Beschreiben Sie zu jedem der 3 Probleme eine mögliche Lösung.
Beispiel: Hinzufügen eines Null-Checks vor Zeile 349.

c) Beschreiben Sie drei Funde des Checker Frameworks anhand des Source Codes: Apache Commons.