OpenSource Tools für den Software Test

Axel Zschutschke[1], Philip Laube[1]
[1]HEITEC AG

Zusammenfassung

Der Workshop gibt einen Überblick über den Testprozess mit OpenSource Tools. Dabei soll der Testprozess selbst (Anforderungen, Testlevel, Visualisierung der Ergebnisse) sowie seine technische Umsetzung dargestellt werden. Die Teilnehmer können die Werkzeuge anhand einfacher Beispiele selbst ausprobieren.

Einleitung

Softwareprodukte werden mit ca. 15-50 Fehlern pro 1000 Zeilen Code ausgeliefert [1]. Diese Fehler sind nicht nur unschön, sondern können auch sehr teuer werden. Durch Testprozesse kann die Anzahl der Fehler stark reduziert und das Vertrauen in die Software gestärkt werden [2]. Im Workshop soll gezeigt werden, wie ein solcher Qualitäts- bzw. Testprozess für Entwicklungsprojekte aufgebaut sein kann und welche (Open-Source) Softwaretools ihn unterstützen können. [1-7]

Vitae

Axel Zschutschke hat 5 Jahre an der TU Bergakademie Freiberg als wissenschaftlicher Mitarbeiter mit OpenFOAM experimentiert, eine Software (C++) für Flammensimulationen entwickelt und die (Linux, OSX) IT des Lehrstuhls gepflegt, von der Nutzerverwaltung im Desktopbereich bis zum Cluster und den CI Werkzeugen. Seit 3 Jahren ist er bei der Firma HEITEC als Entwickler und Testverantwortlicher für die Teststrategie, Testplanung und Pflege der Testumgebungen verantwortlich.

Philip Laube hat während seines Informatikstudiums seinen Schwerpunkt auf Datenbanken und Datenbankprogrammierung gelegt. Mit diesem Wissen hat er dann bei einem lokalen namenhaften Automobilhersteller die Infrastruktur des lokalen Rechenzentrums gepflegt. Seit 2 Jahren ist er bei der Firma HEITEC als Tester und Entwickler tätig. Besonderen Fokus legt er auf die Pflege der CI Strecke und Automatisierung der Testumgebung.

Vorkenntnisse

Der Workshop richtet sich eher an Einsteiger im Thema, es soll ein Überblick über den Testprozess und verschiedene Werkzeuge erarbeitet werden, daher müssen natürlich Grenzen bei der inhaltlichen Tiefe gezogen werden. Ein wenig Programmiererfahrung (idealerweise mit C/C++) und Kenntnisse der wichtigsten Programmierwerkzeuge (Editor, git,...) sollten mitgebracht werden.

Inhalte

Softwarequalität ist nicht nur eine Frage des Tests, sondern auch eine Frage des Entwicklungsprozesses - daher wird sich der Workshop in drei Teile gliedern. Im ersten Teil wird der Entwicklungsprozess vorgestellt und gezeigt an welchen Stellen der Qualitätsprozess mit diesem interagiert. Im zweiten soll anhand einfacher Codebeispiele gezeigt werden, welche Arten von Analysewerkzeugen und Test-Frameworks bereitstehen, um die Code-Qualität zu bewerten bzw. zu steigern. Zuletzt, im driten Teil, soll die Integration dieser Werkzeuge im Build-Prozess dargestellt werden, dabei soll auch die Versionierung und Visualisierung der Ergebnisse beschrieben werden. Es gibt hierbei natürlich kein Patentrezept für alle Softwareprojekte. Unterschiedliche Platformen und Programmiersprachen, oder einfach unterschiedliche Anforderungen an die Softwarequalität spielen dabei eine Rolle. So sind Fehler in der Speicherverwaltung bei be-stimmten Sprachen (C/C++) naturgemäß wahrschein-licher, als wenn die Speicherverwaltung durch die Umgebung erfolgt. Ein Hobby-Projekt soll zwar auch in einem Jahr noch Wartbar sein, aber es drohen (meistens) keine Personenschäden, wenn es nicht alle Randbedingungen geprüft hat.

Betrachtete Werkzeuge

Die folgenden Werkzeuge wollen wir uns näher anschauen. Besonderen Fokus legen wir zum einen auf einfache Frameworks, welche dem Entwickler lokal zur Verfügung stehen und zum anderen auf die Austauschformate der erhaltenen Daten - denn die Visualisierung der Ergebnisse ist mitunter wichtiger als die Ergebnisse selbst.

- googletest (unit test framework)
- clang-tools (formatter, analyzer, coverage)
- gcc, gcov, lcov (compiler + coverage)
- cppcheck (static code analysis)
- valgrind (memcheck, profiler)
- cmake, ctest, cdash (build-chain + test visualization)
- cccc (code metrics)
- git (version control)
- jenkins (CI pipeline)
- sonarqube (as dashboard, C++ static analysis plugin is commercial)
- python3 (modules nach Interesse/Bedarf/Zeit)
- Ausblick cloud: github, travis-CI, coveralls, ...

Quellen

[1] S. McConnel, Code Complete (2013)
[2] S. Grünfelder, Software-Test für Embedded Systems (2017)
[3] A. Spillner, T. Linz, Basiswissen Softwaretest (2012)
[4] J. Bach, The Future of the Testing Role (2017) https://www.youtube.com/watch?v=c5821YeWico
[5] P. Zimmerer, The Future of Testing is Continuous, OOP-Konferenz (2019)
[6] R. Osherove, The Art od Unit Testing (2015)
[7] A. Axelrod, Complete Guide to Test Automation (2018)