Softwaretests

Christoph Betschart

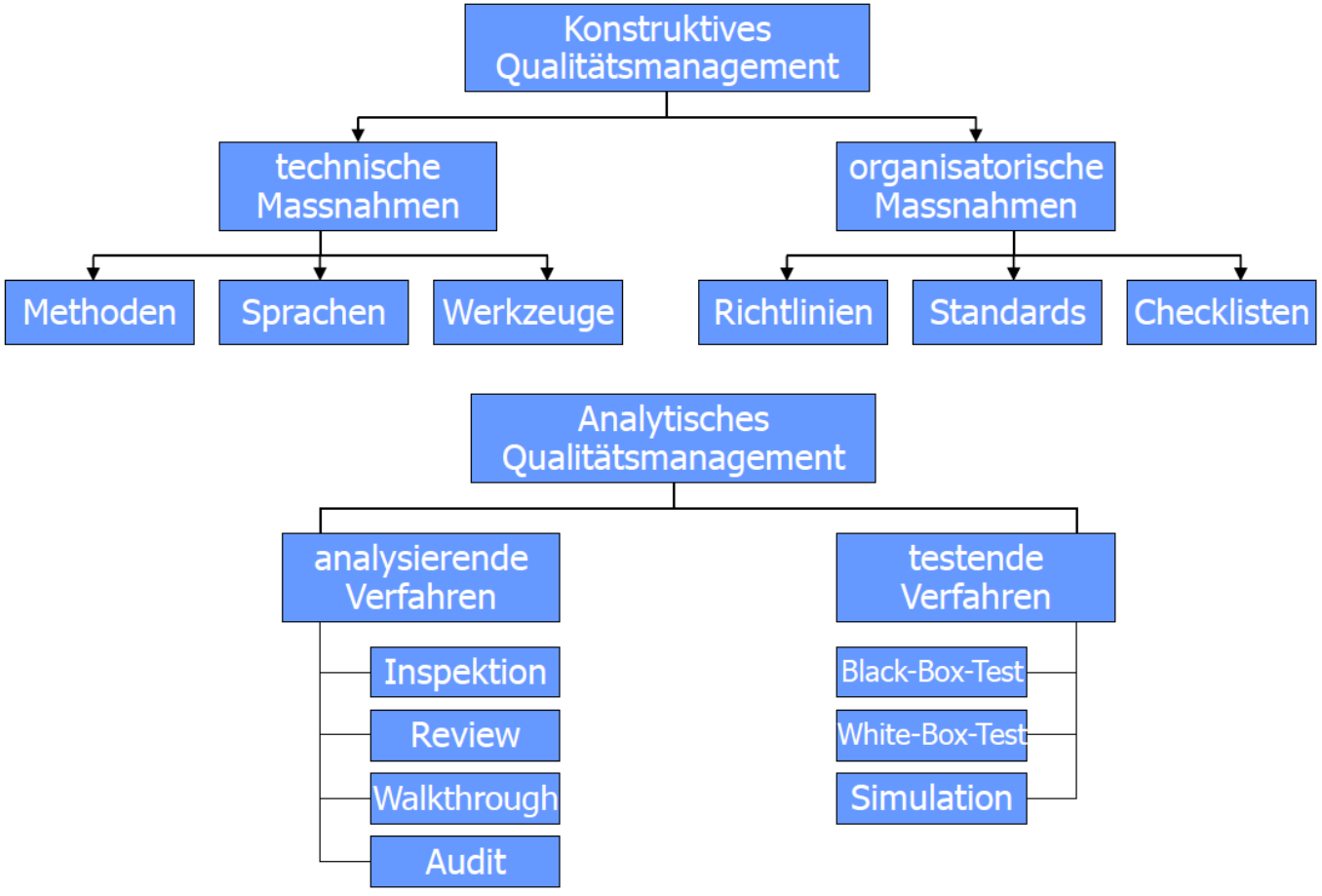
27. Oktober 2014

Inhaltsverzeichnis

Motivation

- Fehler in einem Softwaresystem kann viel Geld kosten.
- Softwaretests als Teil der Qualitätssicherung.
- Softwaretests sollen Fehler in Softwaresystemen frühzeitig erkennen.
- Softwaretests zur Erhöhung der Quellcodequalität

Einordnung



Qualitätsmanagement

Softwaretests

Testfall (Test Case)

Beschreibung eines Tests

- Testumgebung
- Eingabewerte
- Testprozedur
- Erwartete Ausgabewerte (auch Exceptions)

Eingabewerte

- typische Werte
- Grenzwerte
- fehlerhafte Werte

Automatisierte Ausführung von Softwaretests

Softwaretests

Testfall (Test Case)

Beschreibung eines Tests

- Testumgebung
- Eingabewerte
- Testprozedur
- Erwartete Ausgabewerte (auch Exceptions)

Eingabewerte

- typische Werte
- Grenzwerte
- fehlerhafte Werte

Automatisierte Ausführung von Softwaretests

Testing und Programmverifikation

Edsger W. Dijkstra, 1972:

„Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence.“

- Softwaretests decken nur wenige Beispiele aus einer riesigen Menge von möglichen Inputs ab.
- Programmverifikation beweist mathematisch die Korrektheit eines Programmes.

Arten von Softwaretests

Testart	**Was wird getestet?**
Unit Tests	Einzelne Module
Akzeptanztests	Erfüllung der Anforderungen
Usability-Test	Workflows
Regressionstest	Altes Verhalten vs. neues Verhalten
Performancetest	Ressourcenverbrauch
Integrationtest	Zusammenarbeit verschiedener Module
Systemtest	Gesamtsystem
GUI-Tests	Verhalten des GUI

- Aufteilung der Tests führt zu Reduktion der Komplexität der jeweiligen Prüfaktivität.

Seven Principles of Software Testing

Artikel von Professor Bertrand Meyer, ETH Zürich

Principle 1: Definition

To test a program is to try to make it fail.

Beim Testen geht es darum Fehler zu provozieren/finden, nicht diese zu beheben.

Principle 2: Tests versus specs

Tests are no substitute for specifications.

Spezifikationen abstrahieren und generalisieren das Problem, wobei Tests jeweils individuelle Fälle abdecken.

Principle 3: Regression testing

Any failed execution must yield a test case, to remain a permanent part of the project's test suite.

Überprüfe, ob Fehler, welche behoben wurden, zu einem späteren Zeitpunkt wieder auftreten. Vergleiche das Verhalten zum Verhalten einer früheren Version, welches als korrekt eingestuft wurde. Fehler die früher mal behoben wurden, sollen nicht mehr auftreten.

Principle 4: Applying oracles

Determining success or failure of tests must be an automatic process (..) consisting of monitoring contract satisfaction during execution.

Ein Test welcher sagt, es ist „vielleicht“etwas schief gegangen bringt nichts. Eine Überprüfung vom Tester kostet Zeit. Spezifikationen und Verträge (z.B. zwischen Modulen) können genutzt werden, um einen klaren Soll-Zustand zu formulieren.

Principle 5: Manual and automatic test cases

An effective testing process must include both manually and automatically produced test cases.

Nicht die Ausführung betreffend. Automatisch generierte Tests sind gut in der Breite, viele verschiedene Eingabewerte und Parameter welche der Mensch evtl. vergessen würde. Manuell generierte Tests sind gut in der Tiefe. Nutzen das Wissen des Programmierers, welcher das Problem und die Datenstrukturen kennt.

Principle 6: Empirical assessment of testing strategies

Evaluate any testing strategy, however attractive in principle, through objective assessment using explicit criteria in a reproducible testing process.

Wähle die Teststrategie, welche am meisten Nutzen bringt. Auch Teststrategien die wenig bis kein Wissen über das Problem nutzen, können Fehler aufdecken.

Principle 7: Assessment criteria

A testing strategy's most important property is the number of faults it uncovers as a function of time.

Der Nutzen einer Strategie wird über die Anzahl aufgedeckter Fehler pro Zeit gemessen.

Continuous Integration

- Ziel: Steigerung der Softwarequalität
- Erstellung eines Builds und Ausführung automatisierter Tests geschieht bei jedem Einchecken in die Versionverwaltung
- Profitiert von häufigen Commits
- Es gibt immer einen Build zum Testen

Travis-CI

- CI-Tool für GitHub Projekte.
- Unterstützt Runtimes verschiedener Sprachen und verschiedener Versionen von Haus aus.
- Läuft in VM, benötigte Tools sind installierbar.

Build Matrix

Mehrere Builds: Alle Kombinationen der Konfigurationsparameter werden getestet.

Demo:

- Buildbot example:
  `http://buildbot.fast-downward.org/grid`
- Travis-CI: `https://travis-ci.org/`

Tests in FLOSS-Projekten

Grosse Projekte haben eigene Testtools:

- Gecko-basierte Anwendungen: Mozmill
- Ubuntu: Bsp. TestDrive

Quellen

- Artikel: Seven Principles of Software Testing, Professor Bertrand Meyer, ETH Zürich
- Vorlesung: Software Engineering, Franz-Josef Elmer
- Vorlesung: CS108 Programmierprojekt, Software Qualitätssicherung
- Travis-CI: `http://docs.travis-ci.com/user/getting-started/`