Application Testing

Professorship of Open Source Software

Friedrich-Alexander University Erlangen-Nürnberg

ADAP B03

Software Defects

- Software is omnipresent in our daily life
- Bugs and errors in software can have grave consequences

Remember the Mars Climate Orbiter incident from 1999?

https://slideplayer.com/slide/5829764/19/images/37/Remember+the+Mars+Climate+Orbiter+incident+from+1999.jpg

FEHLER IN DER SOFTWARE

Zehntausenden Studenten droht Bafög-Verspätung

AKTUALISIERT AM 24.08.2016 - 16:08

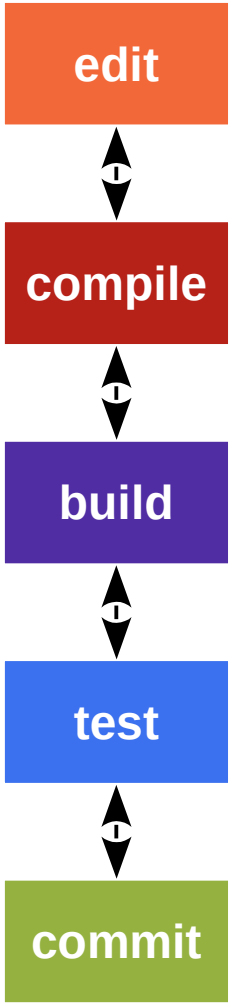
Seit dem ersten August gibt es höhere Bafög-Sätze und Freibeträge. Doch eine gängige Bearbeitungs-Software kennt diese Neuerungen nicht. Deshalb könnten viele Studenten zum Semesterstart erst einmal ganz ohne Geld dastehen.

https://www.faz.net/aktuell/karriere-hochschule/campus/fehler-in-der-software-zehntausenden-studenten-droht-bafoeg-verspaetung-14403775.html

Why Testing Anyway?

- Find failures and defects and prevent them reaching the production version
- Check if item under test works as one expects
- Reduce level of risk of inadequate software quality
- Check if requirements have been satisfied
- Gain confidence in the quality of the item under test
- Comply with legal or contractual requirements or standards

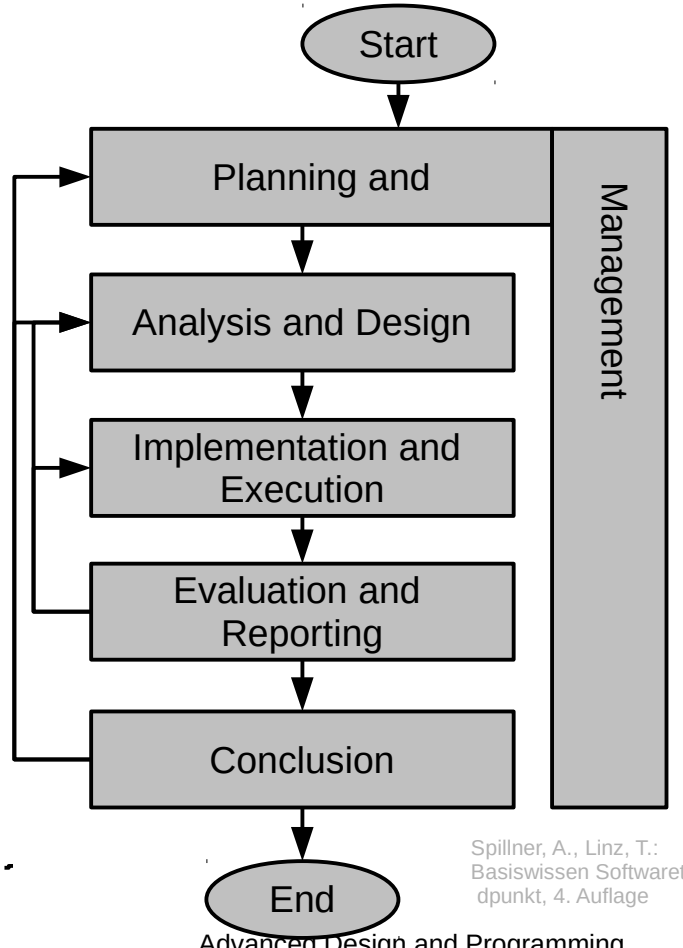
Simple Development Cycle



- **Edit** = developer implements new feature
  - Iterates over the code until it looks right
- **Compile** = developer compiles the code
  - Iterates over the code until it compiles (no syntax error)
- **Build** = developer puts classes, build path together
  - Packages application
- **Test** = developer tests the program
  - Keeps going until “behavior looks right” i.e. no bugs
- **Commit** = developer commits to code repository
  - May trigger a CI pipeline

Testing is a Process

- In larger projects testing needs to be governed by a process
  - Planning and managing the test process
  - Analysing which tests are necessary and designing them
  - Implementing the tests and executing them
  - Evaluation of test results and reporting
  - Learn for the future



Spillner, A., Linz, T.:
Basiswissen Softwaretest,
dpunkt, 4. Auflage

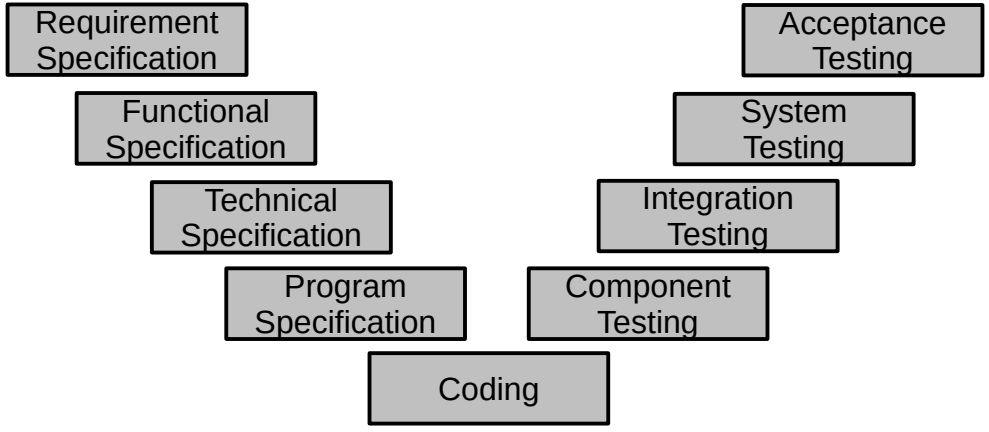
Static vs. Dynamic Testing

- **Static testing**
  - No execution of the software necessary
  - Static code analysis
  - Reviews
    - of code
    - of diagrams
    - of documents, e.g. requirement specification

- **Dynamic Testing**
  - Testing at run-time of a software

Test Levels

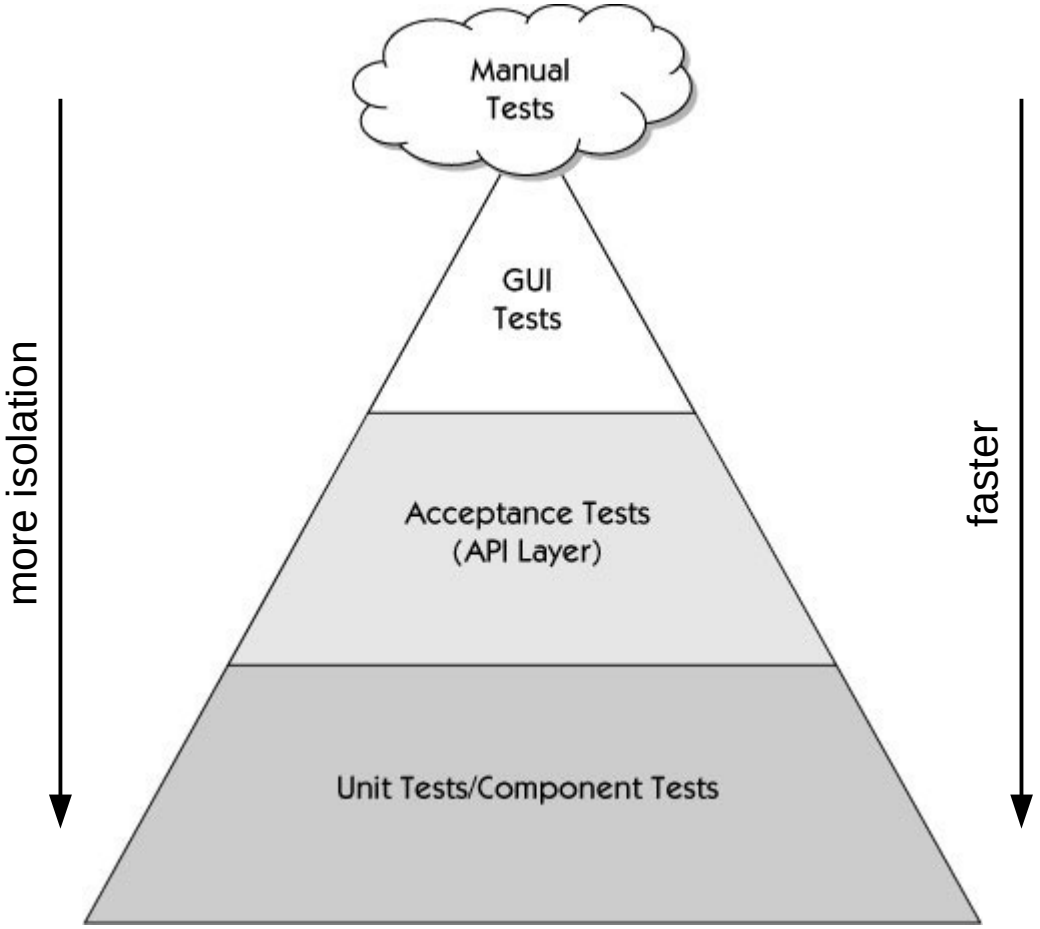
- **Components tests** (a.k.a. unit tests)
  - Focus on testing one component out of context
- **Integration tests**
  - Focus on the collaboration of different components
- **System tests**
  - Focus on the system as a whole
- **Acceptance tests**
  - Focus on customer and end user experience



Test Types

- **Functional testing** (a.k.a. **Blackbox Testing**)
  - Tests against functional requirements
  - Tests the behaviour that is visible to the outside world
- **Non-functional testing**
  - Tests against non-functional requirements
  - E.g. performance, stress, usability, portability requirements
- **Structure-based testing** (a.k.a. **Whitebox Testing**)
  - Tests against internal structure of component or system
  - Goal is to cover all elements of the spectated structure
- **Testing related to changes**
  - Retests after a bug was fixed to ensure the fix
  - Regression testing after any changes on the software

Test Automation Pyramid



- Try to automate as much as possible
- Manual testing
  - takes time that can be saved by automation
  - is not as reliable as programmed tests
  - tends to be selective, not comprehensive
- But:
  - Human intuition can see problems that computers cannot
  - Some things can't be automated
    - e.g. usability tests

Janet Gregory, Lisa Crispin: Agile Testing – A Practical Guide for Testers and Agile Teams (2008)

7 Software Testing Principles

1) Testing shows the presence of defects
   - “Program testing can be used to show the presence of bugs, but never to show their absence!” – Edsger W. Dijkstra, 1970
2) Exhaustive testing is not possible
   - E.g. input field for emails: can we test each input?
3) Start testing early
   - The earlier we find defects (e.g. in the requirement analysis phasis) the less the costs for fixing them
4) Testing is context-dependent
   - Safety-critical systems are differently tested than apps!
5) Defect clustering
   - Pareto principle: 80% of problems are found in 20% of the modules
6) Pesticide paradoxon
   - Just retrying tests has no benefits. Test cases have to be reviewed and revised.
7) Absence of error
   - If tests find no defects, there still might be some in the system!

Testing Terminology

- **Test (Case)**
  - A single test for some particular aspect of the software, succeeds or fails
- **Test Suite**
  - A set of related tests that cover a particular domain of the software
- **Test Set-up**
  - The data and preparation necessary to run a test as intended
- **Test Result**
  - The result of running a test, typically succeeds/fails or error
- **Test Harness**
  - A software, like JUnit, that is used to run test suites

Thank you! Questions?

dirk.riehle@fau.de – http://osr.cs.fau.de

dirk@riehle.org – http://dirkriehle.com – @dirkriehle

Credits and License

- Original version
  - © 2020 Friedrich-Alexander University Erlangen-Nürnberg and Dirk Riehle, all rights reserved
- Contributions
  - Georg Schwarz (2019)