Software Testing

Lesson 5 – Static Testing

Uwe Gühl

Winter 2015 / 2016

Contents

- Static Testing
  - Static Techniques and the Test Process
  - Review Process
  - Static Analysis by Tools

Static Techniques and the Test Process

- Dynamic testing techniques
  ⇨ requires the execution of software
- Static testing techniques
  ⇨ without execution of software
  ⇨ early test activity
  - Reviews
    Manual examination of the code or other project documentation (tool support possible)
  - Static analysis
    Automated analysis of the code

Static Techniques and the Test Process

- The main review manual activity is to examine a work product and make comments about it, e.g.
  - Requirements specifications
  - Design specifications
  - Code
  - Test plans
  - Test specifications
  - Test cases
  - Test scripts
  - User guides
  - Web pages

Static Techniques and the Test Process

- Benefits of reviews
  - Early defect detection and correction
  - Development productivity improvements
  - Reduced development time-scales
  - Reduced testing cost and time
  - Lifetime cost reductions
  - Fewer defects and improved communication
- Reviews can find missing items, for example, in requirements, which are unlikely to be found in dynamic testing.

Static Techniques and the Test Process

- Reviews, static analysis and dynamic testing have the same objective – identifying defects
- They are complementary
  Different techniques can find different types of defects effectively and efficiently
  - Static techniques: Find defects – causes of failures
  - Dynamic testing: Find failures

Static Techniques and the Test Process

- Typical defects that are easier to find in reviews than in dynamic testing
  - Deviations from standards
  - Requirement defects
  - Design defects
  - Insufficient maintainability
  - Incorrect interface specifications

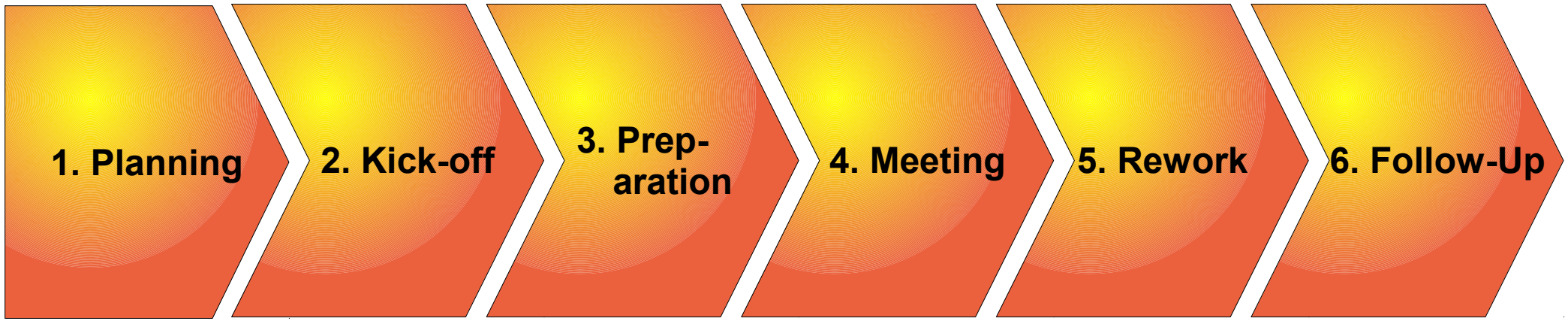
Review Process

- Types of reviews
  - **informal**, characterized by
    no written instructions for reviewers.
  - **systematic**, characterized by
    - team participation
    - documented results of the review
    - documented procedures for conducting the review
- The formality of a review process is related to
  - maturity of the development process
  - any legal or regulatory requirements
  - the need for an audit trail

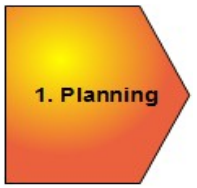
Review Process

- The way a review is carried out depends on the agreed objectives of the review, for example
  - find defects
  - gain understanding
  - educate testers and new team members
  - discussion and decision by consensus

Review Process
Activities of a Formal Review



Review Process
Activities of a Formal Review



1. Planning

- Defining the review criteria
- Selecting the personnel
- Allocating roles
- Defining the entry and exit criteria for more formal review types (e.g., inspections)
- Selecting which parts of documents to review
- Checking entry criteria (for more formal review types)

Review Process
Activities of a Formal Review

2. Kick-off

- Distributing documents
- Explaining the objectives, process and documents to the participants

Review Process
Activities of a Formal Review

3. Individual preparation

- Preparing for the review meeting by reviewing the document(s)
- Noting potential defects, questions and comments

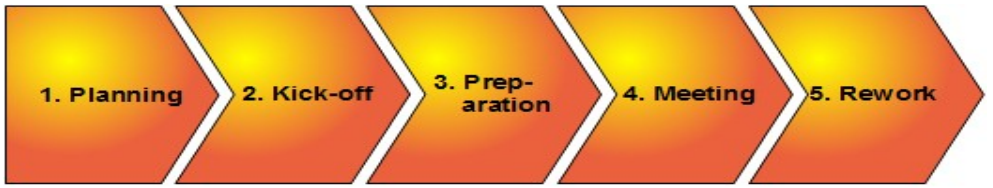
Review Process
Activities of a Formal Review

4. Review meeting

… to examine / evaluate / record results

- Discussing or logging, with documented results or minutes (for more formal review types)
- Noting defects, making recommendations regarding handling the defects, making decisions about the defects
- Examining / evaluating and recording issues during any physical meetings or tracking any group electronic communications

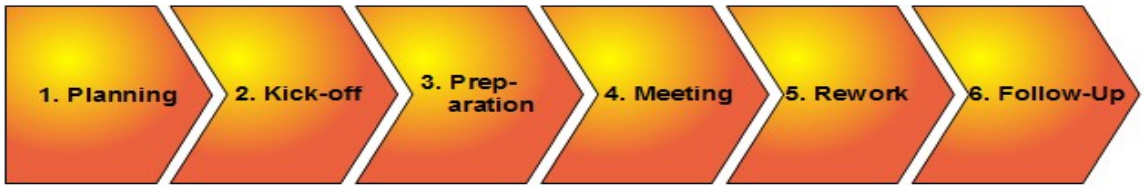
Review Process
Activities of a Formal Review



5. Rework

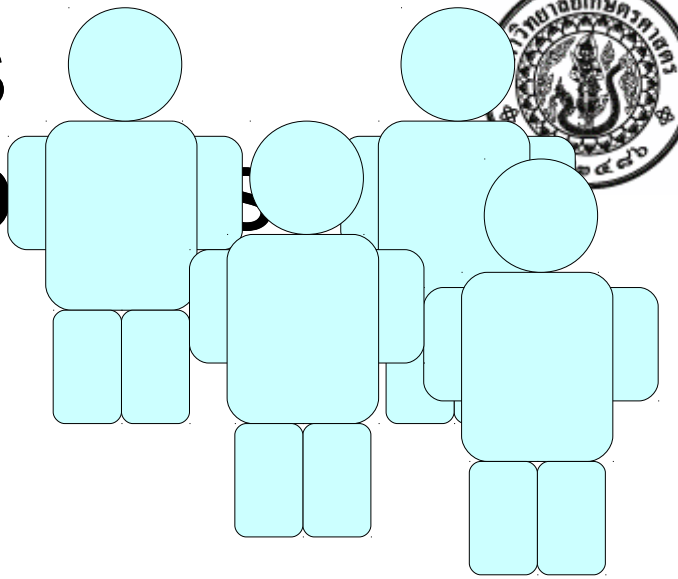
- Fixing defects found (typically done by the author)
- Recording updated status of defects (in formal reviews)

Review Process
Activities of a Formal Review

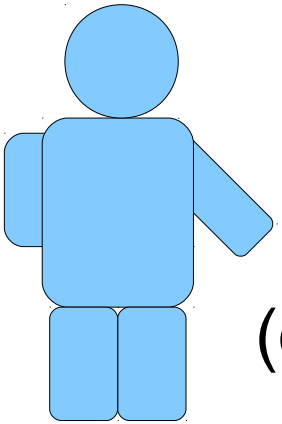


6. Follow-up
   - Checking that defects have been addressed
   - Gathering metrics
   - Checking on exit criteria (for more formal review types)

Review Process Roles and Responsib

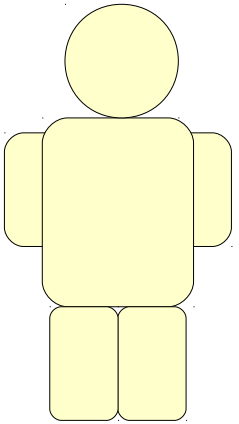


Overview

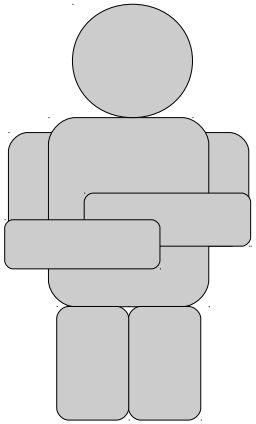


Moderator

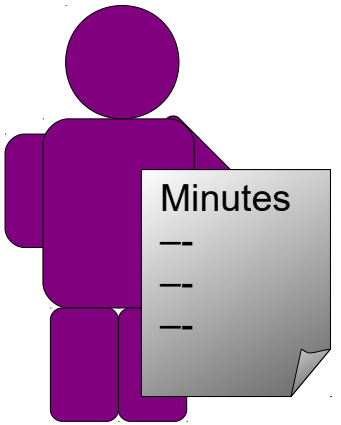
Reviewers
(or checkers, inspectors)



Author



Manager

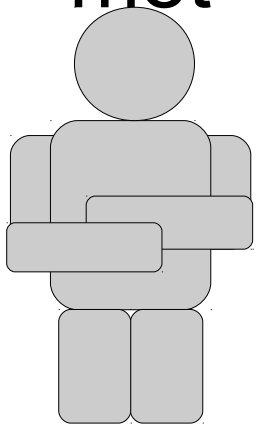


Scribe
(or recorder)

Review Process
Roles and Responsibilities

Manager

- decides on the execution of reviews
- allocates time in project schedules
- determines if the review objectives have been met

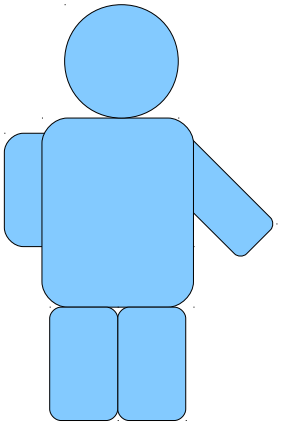


Manager

Review Process
Roles and Responsibilities

Moderator

- leads the review of the document(s), including
  - planning the review
  - running the meeting
  - following-up after the meeting



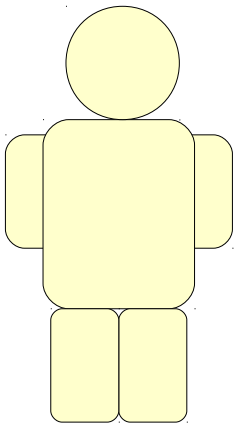
Moderator

- mediates between the various points of view, if necessary
- is often the person upon whom the success of the review rests

Review Process
Roles and Responsibilities

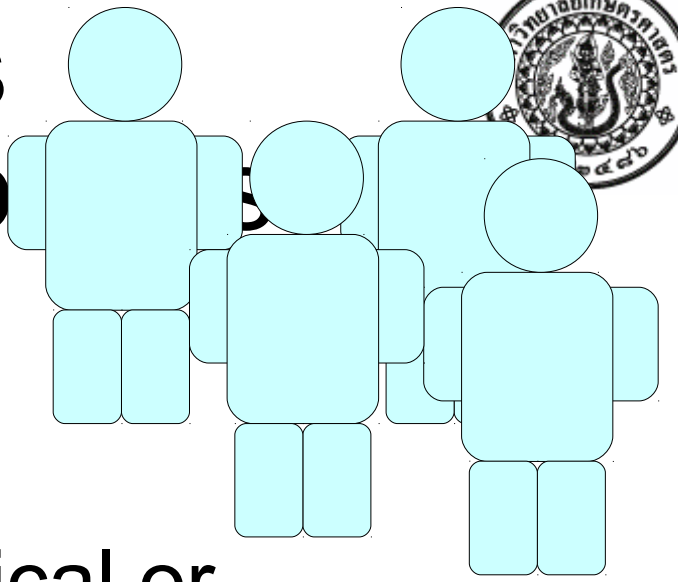
Author

- Writer or person with chief responsibility for the document(s) to be reviewed



Author

Review Process Roles and Responsibilities



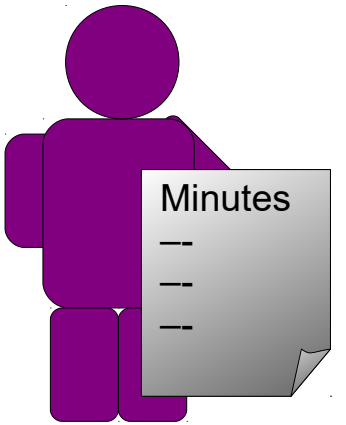
Reviewers
Synonyms: Checkers, inspectors

- Individuals with a specific technical or business background who, after the necessary preparation, identify and describe findings (e.g., defects) in the product under review
- Reviewers should
  - be chosen to represent different perspectives and roles in the review process
  - take part in any review meetings

Review Process
Roles and Responsibilities

Scribe
Synonyms: Recorder, minute taker

- documents all the items identified during the meeting like
  - issues
  - problems
  - open points



Scribe
(or recorder)

Review Process
Types of Reviews

- Informal Review
- Walk-through
- Technical Review
- Inspection

Could be performed as a “Peer Review” by colleagues of the producer of the product

Review Process
Types of Reviews

- A single software product or related work product may be the subject of more than one review
- If more than one type of review is used, the order may vary, examples:
  - Informal review before a technical review
  - Inspection on a requirements specification before a walk-through with customers

Review Process
Informal Review

- No formal process
- May take the form of
  - pair programming
  - a technical lead reviewing designs and code
- Results may be documented
- Varies in usefulness depending on the reviewers
- Main purpose:
  Inexpensive way to get some benefit

Review Process Walk-through (1/2)

- Meeting led by author
- May take the form of
  - scenarios
  - dry runs
  - peer group participation
- Open-ended sessions
  - Optional pre-meeting preparation of reviewers
  - Optional preparation of a review report including list of findings

Review Process
Walk-through (2/2)

- Optional scribe, who is not the author
- May vary in practice from quite informal to very formal
- Main purposes
  - Learning
  - Gaining understanding
  - Finding defects

Review Process
Technical Review (1/3)

- Documented, defined defect-detection process that includes peers and technical experts with optional management participation
- Ideally led by trained moderator (not the author)
- Pre-meeting preparation by reviewers requested

Review Process
Technical Review (2/3)

- Optional use of checklists
- Preparation of a review report could include
  - list of findings
  - an evaluation if the software product meets its requirements
  - recommendations related to findings
- Could vary from quite informal to very formal

Review Process
Technical Review (3/3)

- Main purposes
  - Discussing
  - Making decisions
  - Evaluating alternatives
  - Finding defects
  - Solving technical problems
  - Checking conformance to
    - specifications
    - plans
    - regulations
    - standards

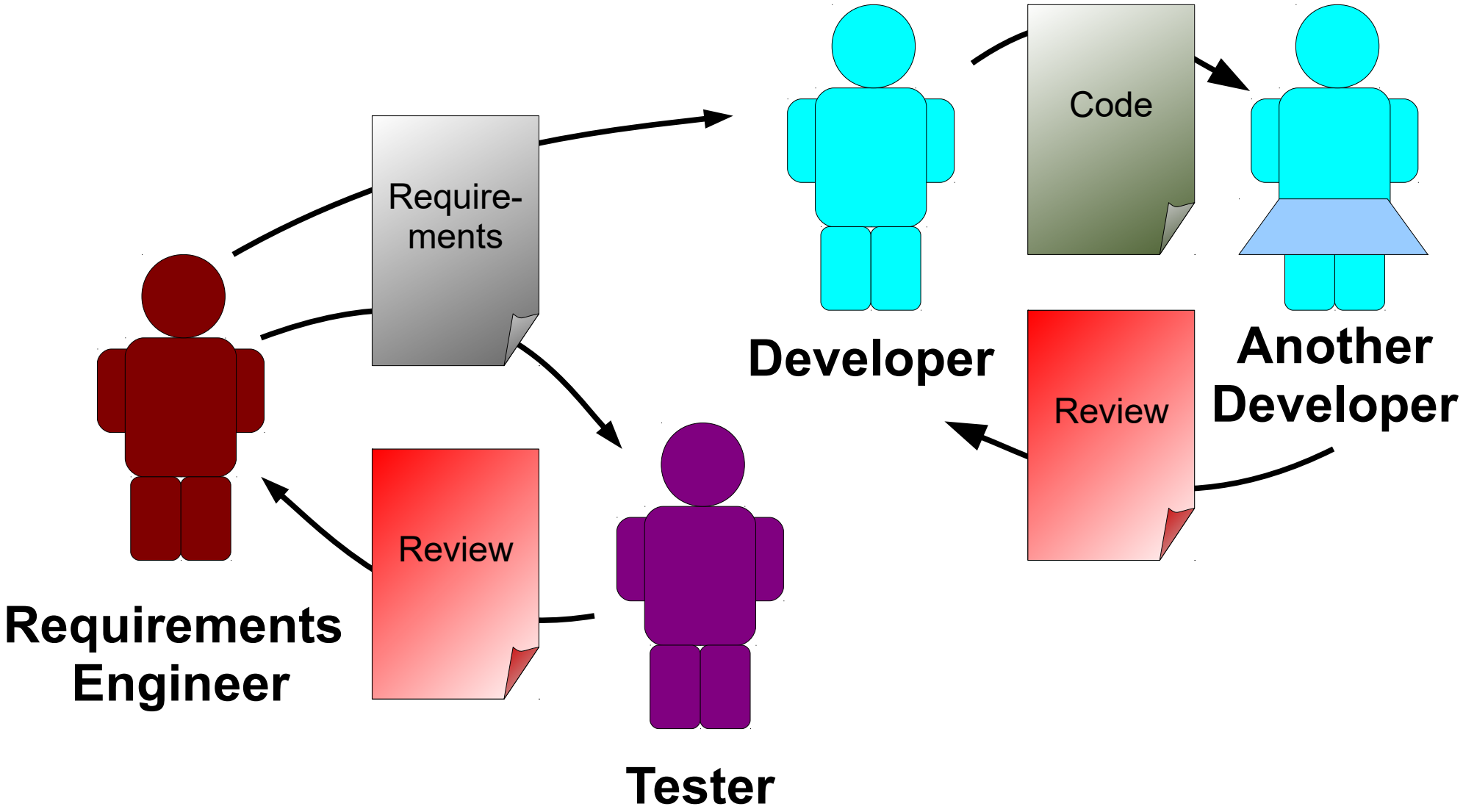
Review Process Inspection (1/2)

- Main characteristics
  - Led by trained moderator (not the author)
  - Usually conducted as a peer examination
  - Defined roles
  - Includes metrics gathering
  - Formal process based on rules and checklists
  - Specified entry and exit criteria for acceptance

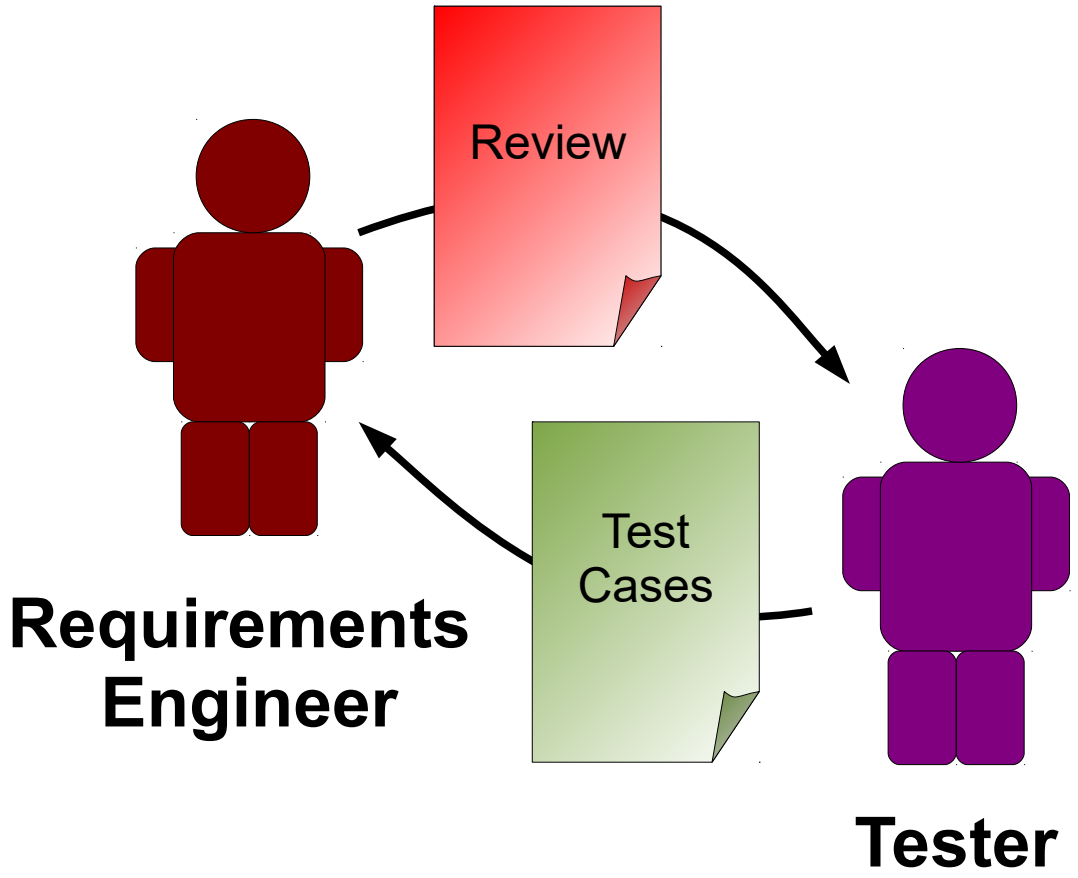
Review Process
Inspection (2/2)

- Main characteristics
  - Pre-meeting preparation
  - Inspection report including list of findings
  - Formal follow-up process
    … with optional process improvement components
  - Optional reader
- Main purpose: Finding defects

Review Process Example (1/2)



Review Process Example (2/2)



Review Process Success factors (1/3)

- Checklists, for example
  - based on various perspectives such as user, maintainer, tester or operations
  - typical requirements problems
- Attitudes
  - Emphasis on learning and process improvement
  - Defects found are welcomed and expressed objectively
  - People issues and psychological aspects are dealt with; e.g., making it a positive experience for the author
  - Atmosphere of trust: The outcome will not be used for the evaluation of the participants

Review Process
Success factors (2/3)

- Each review has clear predefined objectives
- Right people for the review objectives are involved
- Testers are valued reviewers who
  - contribute to the review
  - learn about the product
    which enables them to prepare tests earlier

Review Process
Success factors (3/3)

- Training is given in review techniques, especially in the more formal techniques
- Management supports a good review process; e.g. by incorporating adequate time for review activities in project schedules

Review Process
Cost-value ratio (1/2)

- Reviews cost about 10 to 15 % of development budget
- Reviews save costs [Bus90] [FLS00] [GG96]:
  - About 14% up to 25% savings in IT projects possible (additional costs of reviews already considered)
  - It's possible to find up to 70% of defects in a document
  - Reduction of defect costs up to 75%

Review Process
Cost-value ratio (2/2)

- „Peer reviews“ – capable experts review the work
  Use: *will detect about 31 % up to 93 % of all defects, average: 60 %*
- “Perspective review” – evaluators use the work for own tasks
  Use: *35 % more defects are detected compared to non-purposeful reviews*
  Example: Review of a specification:
  - Tester: … to generate test cases out of it
  - Documentation: … to write an user manual out of it

Static Analysis by Tools

- The objective of static analysis is to find defects in software source code and software models
- Distinguish
  - Static analysis is performed without actually executing the software being examined by the tool
  - Dynamic testing does execute the software code
- Static analysis tools analyse program code (e.g., control flow and data flow), as well as generated output such as HTML and XML

Static Analysis by Tools Value

- Early detection of defects prior to test execution
- Early warning about suspicious aspects of the code or design by the calculation of metrics, such as a high complexity measure
- Identification of defects not easily found by dynamic testing
- Detecting dependencies and inconsistencies in software models such as links
- Improved maintainability of code and design
- Prevention of defects,
  if lessons are learned in development

Static Analysis by Tools
Typical defects discovered (1/2)

- Referencing a variable with an undefined value
- Inconsistent interfaces between modules and components
- Variables that are not used or are improperly declared
- Unreachable (dead) code
- Missing and erroneous logic (potentially infinite loops)

Static Analysis by Tools
Typical defects discovered (2/2)

- Overly complicated constructs
- Programming standards violations
- Security vulnerabilities
- Syntax violations of code and software models

Static Analysis by Tools Usage

- Static analysis tools are typically used
  - by developers (checking against predefined rules or programming standards)
    - before and during component and integration testing
    - when checking-in code to configuration management tools
  - by designers
    - during software modelling
- Compilers may offer some support for static analysis, including the calculation of metrics

Static Analysis by Tools
Data flow analysis

- Analysis data flow in the code to find anomalies (=> These could cause failures)
- Anomaly [IEEE 1044]
  Any condition that deviates from expectation based on requirements specifications, design documents, user documents, standards, etc., or from someone's perception or experience.

Static Analysis by Tools
Data flow analysis

- For every variable there is a status defined
  - d = defined
    The variable gets defined.
    A value gets assigned, the variable has a value.
  - r = referenced
    The variable gets read or is used.
  - u = undefined
    The variable has no defined value.

Static Analysis by Tools
Data flow analysis

- Anomalies
  - dd (defined / defined)
    Defined, then gets defined again before first value gets used
  - du (defined / undefined)
    Defined, then gets invalid or undefined without use
  - ur (undefined / referenced)
    Undefined variable read or used

Static Analysis by Tools
Data flow analysis

- Anomalies
  - dd

```
int x = function1();
x = function2();        // redefinition of x -> dd
```

  - du

```
{
  int x = 2;
}                       // x undefined at exit -> du
```

  - ur

```
int x;                          // x undefined
int y = x;                      // x referenced → ur
```

Static Analysis by Tools
Data flow analysis

Example: Function `MinMax` should sort 2 numbers

```
void MinMax(int& Min, int& Max)
{
    int Help;
    if (Min > Max)
    {
        Max = Help;
        Max = Min;
        Help = Min;
    }
}
```

Help	Min	Max
	d	d
u		
	r	r
r		d
	r	d
d	r	
u		

ur Anomaly

dd Anomaly

du Anomaly

Static Analysis by Tools Overview

- [Cat16] and [Wik16] list tools for static code analysis for different program languages
- 4 static analysis tools for Java have been compared [AKG+10].
  Result:
  - Jtest has had the highest defection ratio
  - Findbugs as open source tool was second

  Advice from the authors:
  - Take the respective advantage of several tools for detecting bugs in different categories

Sources

- [AKG+10] Md. Abdullah Al Mamun, Aklima Khanam, Håkan Grahn, and Robert Feldt: Comparing Four Static Analysis Tools for Java Concurrency Bugs, 2010, http://robertfeldt.net/publications/grahn_2010_comparing_static_analysis_tools_for_concurrency_bugs.pdf
- [Bus90] Bush, M.: Software Quality: The use of formal inspections at the Jet Propulsion Laboratory. In: Proc. 12th ICSE, p. 196-199, IEEE 1990
- [CAT16] Source Code Analysis Tools, 2016, http://www.codeanalysistools.com/
- [FLS00] [FLS00] Frühauf, K.; Ludewig, J,; Sandmayr, H.: Software-Prüfung: eine Fibel. vdf, Verlag der Fachvereine, Zürich, 4. Aufl. 2000
- [GG96] Gilb, T.; Graham, D.: Software Inspections. Addison-Wesley, 1996
- [IEEE 1044] IEEE 1044:1993. Standard Classification for Software Anomalies
- [ISTQB-CTFLS11] International Software Testing Qualifications Board: Certified Tester Foundation Level Syllabus, Released Version 2011, http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html
- [ISTQB-GWP15] Glossary Working Party of International Software Testing Qualifications Board: Standard glossary of terms used in Software Testing, Version 3.01, 2015, http://www.istqb.org/downloads/glossary.html
- [Wik16] Wikipedia: List of tools for static code analysis, 2016, https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis