Karlsruhe Institute of Technology

Software Engineering II

Prof. Dr. Raffaela Mirandola

Topic 10
Reviews

SASIS – SELF-ADAPTIVE SOFTWARE-INTENSIVE SYSTEMS
KASTEL – INSTITUTE OF INFORMATION SECURITY AND DEPENDABILITY

sasis.kastel.kit.edu

Overview on Today's Lecture

Content

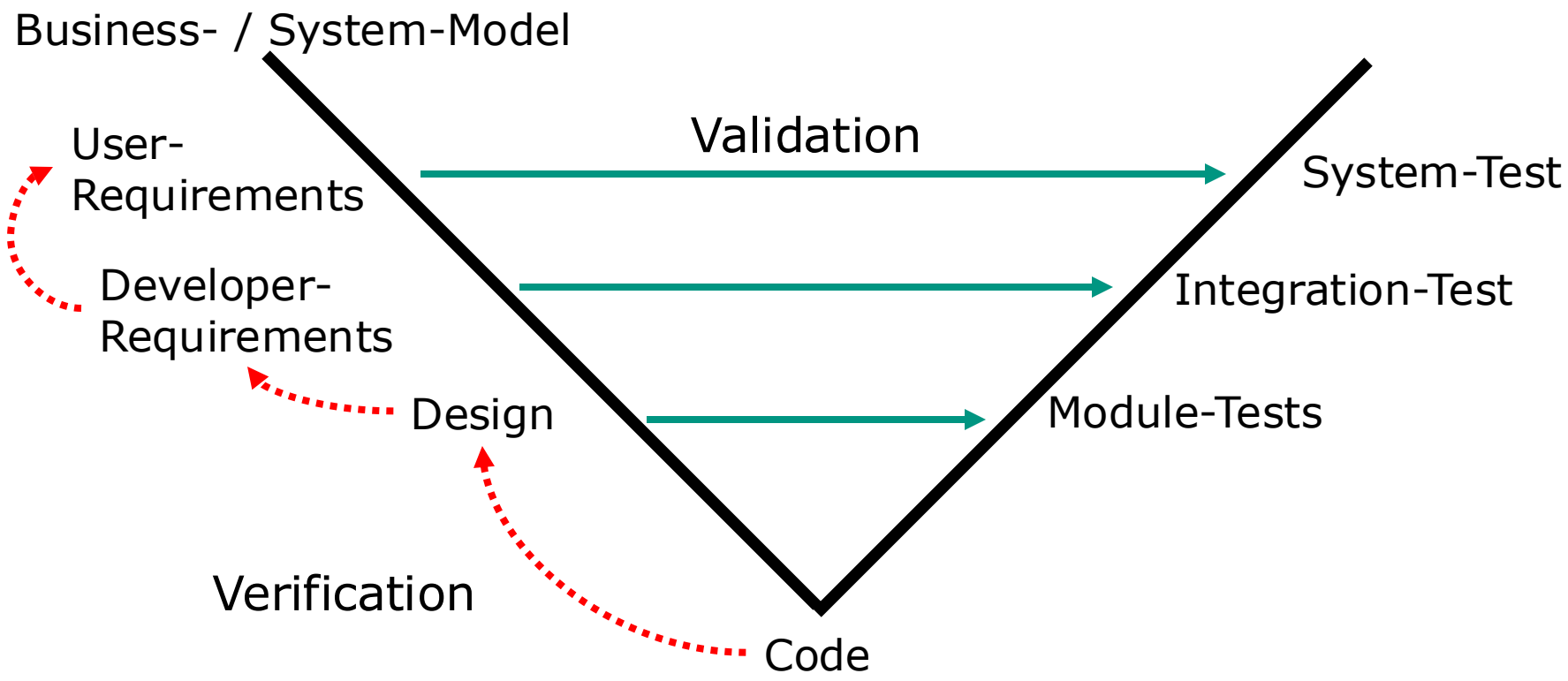
- Motivation
- Benefits of Reviews
- What is a review, walktrhough, etc
- Anti-Patterns hindering good reviews

Learning Goals

- Understand the role and procedure of a review
- Understand the differences between different review appraoches
- Understand the benefits of reviews
- Are aware of the dangers for reviews in form of anti-patterns.

The V-Model

- A model to show different software artifacts and their relations
- Not a process model



Validation and Verification

Informatics Terminology

- **Validation**: case based check on expected behaviour.
- **Verification**: check whether refinement relation holds between two documents:
  - Requirements specification and code
  - Requirements specification and architecture / design
  - Architecture / design and code

Engineering Terminology

- **Validation**: checking whether I build the right product.
- **Verification**: checking whether I build the product right.

Reviews

- *Reviews* are meetings where a software artifact is examined to improve its quality, especially correctness (i.e. to remove errors)
  - Reviews are an informal verification technique (informatics terminology)

- Several specific forms
  - Inspection
  - Team Review
  - Walkthrough
  - Pair Programming
  - Pass-around, Ad-hoc review

- Can concern different artifacts
  - Requirements, project plans, checklists
  - Architecture, design, code, test cases

Reviews are almost a Silver Bullet

- Reports from Glass, Software Facts and Fallacies on Reviews, p. 104

- Rigorous inspections can remove up to 60 – 90 % of errors before testing is started.
  - *Bush, M.: Report at the 14th Annual Software Engineering Workshop, NASA-Goddard, Nov. 1989*
  - *Rifkin, S., Deimel, L.: Applying Program Complehension Techniques to Improve Software Inspections, Software Practitioner, May 1995*
  - *Boehm, B., Basili, V.:Software Defect Reduction Top 10 List, IEEE Computer, Jan. 2001*

They are not often used

- IEEE Software Nov / Dec 2003, pp. 46-51
  Study by Ciolkowski et al. shows state of the practice

- So, if they are nearly a silver bullet, why they are not used more often?
  - consultants cannot make money out of it
  - not new
  - increase of up-front costs
  - requires consideration of psychological factors

Dangers of Reviews – Dangers for Reviews

- **Dangers of reviews**
  - Testing is omitted
  - Authors are frustrated or even feel violated
    - Authors tend to get sloppy
    - „Errors will be found during review anyway.“ (Like with testing, it is not true → cleanroom software-engineering is successful)
    - These dangers will turn into dangers for reviews!

- In addition, there are **dangers for reviews**:
  - Managers cut time if deadline approaches
  - People are not well prepared (waste of time)
  - Obfuscation: Authors tend to protect themselves by tricky code or documents that are difficult to understand

Benefits of Reviews

- Quality, correctness

  Reviews are a verification technique!

- Improved understanding on project
  - Knowledge is shared in a better way
  - Less dependency on availability of one person
- Teaching of style and experience to novices
- Better readability
  - Less tricky code
  - Better documentation / comments

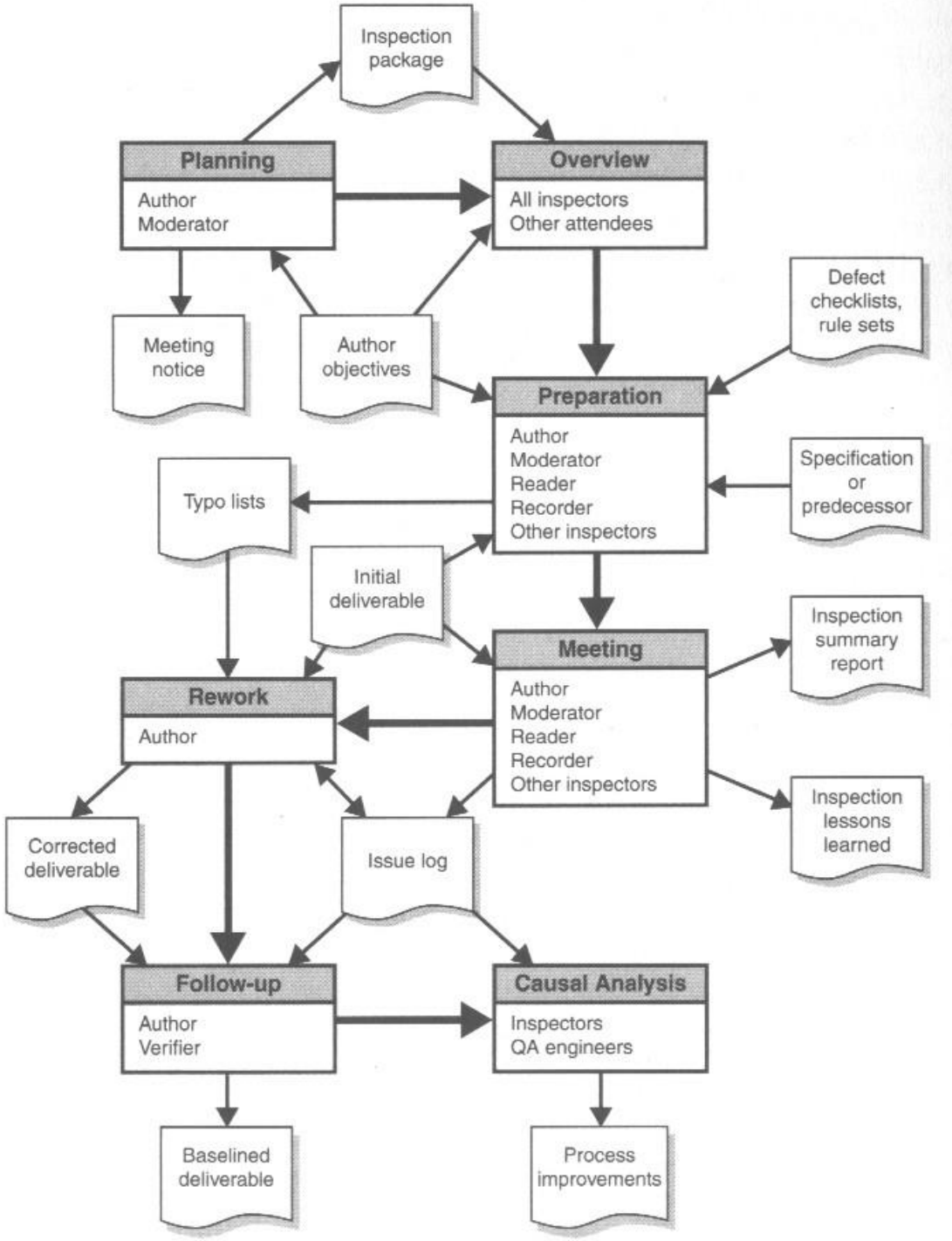
Different Phases of a Review

1. Planning
2. Overview
3. Preparation
4. Meeting
5. Rework
6. Follow-Up
7. Causal Analysis

Roles of a Review

- Author
- Moderator
- Reader(s)
- Recorder
- Verifier

The Inspection Process



Picture from Wiegers [6]

Documents Involved

Document	Comes From	Goes To
Author objectives	Author	Moderator, other inspectors
Meeting notice	Moderator	Inspectors, other attendees
Inspection package	Moderator, Author	Inspectors
Specification (or previous version)	Author of spec.	Inspectors
Typo list	Inspectors	Author
Issue log	Recorder	Author, moderator, verifier
Corrected deliverable	Author	Verifier
Inspection summary report	Moderator	Management
Lessons learned	Moderator	Coordinator of QS
Process Improvements	Inspectors	Process Group

Code Reading Techniques

- Reading based on test case(s)
- Hypothesis-based reading (when seeing a control flow structure): form an assumption what it is for and then check the assumption:
  - **`while`**: array (container iteration), file-read, iterative compution (sum)
  - **`if`**: error handling
  - **`if`** in a **`loop`**: handling of special cases, early loop abortion
- Architecural Reading
  - Quite unclear
  - Detection of patterns through hypotheses

Different Review Types

Review Type	Planning	Preparation	Meeting	Correction	Verification
Inspection	x	x	x	x	x
Team Review	x	x	x	x	
Walkthrough	x		x	x	
Pair Programming	x		continuous	x	x
Peer Deskchecks		x	possibly	x	
ad hoc pass around			x	x	

Psychological Interaction Patterns (1)

The alcoholic pattern

- Roles:
  - Addicted (having a bad habit)
  - Helper
  - Punisher
  - Ashamed
- Addicted: „let‘s see how far I can go“, „see whether you can get me “
- Bad habits during software reviews: late, not prepared, etc.

Psychological Interaction Patterns (2)

The „now-I-have-got-you“ pattern

- Roles:
  - A: discovers that B made a fault
  - B: indeed made a fault
- A uses his or her knowledge to have B at his/her mercy
- B has no other chance than to respond calmly

Psychological Interaction Patterns (3)

The „see-what-you-made-to-me“ pattern

- Roles:
  - A: does a task
  - B: interferes with A
    (interrupt, question, task delegation, ...)
- A makes an error in the performed task because of B‘s interference, A: „See what you have done to me.“
- A producer of document, B reviewer.
- A implements B‘s suggestion (interference) and later something is claimed wrong by A because of this suggestion.

Psychological Interaction Patterns (4)

The „if-it-were-not-for-you“ pattern

- Roles:
  - A: is doing a task and claims towards B: „if it were not for you, I would have achieved <SOME GREAT ACHIEVEMENT>“.
- B cannot fight against it, as A‘s claim is hypothetical
- A developer, B reviewer

Psychological Interaction Patterns (5)

The „Look-how-hard-I-tried“ pattern

- Roles:
  - A: is on a foreseeably failed project (at least for him/her)
  - A shows increasing signs of hard work.
  - With „see, how hard I work“, A aims to persuade B in the future that when the project fails it was not A‘s fault

Psychological Interaction Patterns (6)

The „yes-but“ pattern

- Roles:
  - A proposes improvement to correct error
  - B „yes-but“ player
  - Any constructive help of A is refused by B
    (for the sake of defending his / her „solution“
    or for making his / her error less obvious)
- The only solution:
  - Point B to his/her error and ask B how to improve it
  - → Force B to think constructive.

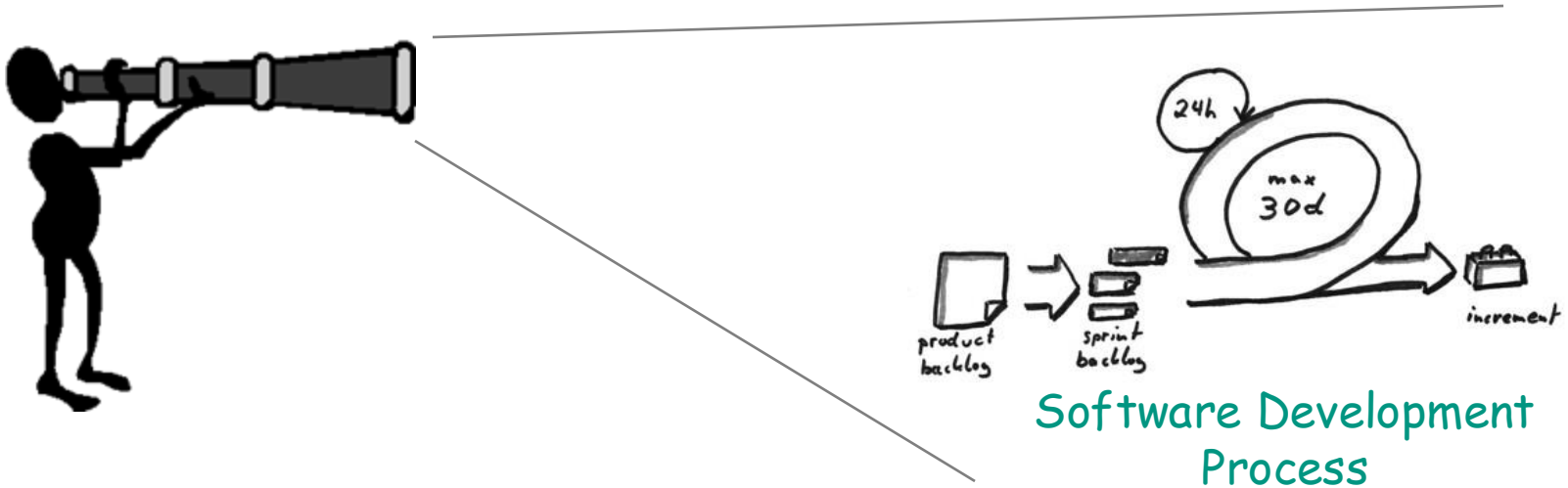
Psychological Interaction Patterns (7)

The „wouldn‘t-it-be-nice-if“ pattern

- Roles:
  - A presents solution
  - B present‘s nice add-on / alternative to improve his / her reputation
- B is usually not constructing systems
- A had to make several compromises
  - because he / she actually constructed a system
- Often: A falls into the yes-but-pattern

Summary on Reviews

- Reviews can heavily increase software quality
- Reviews create up-front project costs
- The success of reviews depends on social and psychological factors



References

- **[1] SDQ Wiki**
  https://sdqweb.ipd.kit.de/wiki
- **[2] Code Review FAQ: mozilla.org's code review process**
  http://www.mozilla.org/hacking/code-review-faq.html
- **[3] Code Review Checklist**
  http://www.macadamian.com/index.php?option=com_content&task=view&id=27&Itemid=31
- **[4] Review-Technik**
  http://www.reviewtechnik.de
- **[5] A Guide to Code Inspections**
  http://ganssle.com/Inspections.pdf
- **[6] Karl E. Wiegers, "Peer Reviews in Software", Addison Wesley, 2002**