3. Static Techniques

Hans Schaefer
hans.schaefer@ieee.org
Http://home.c2i.net/schaefer/testing.html

Reviews
Static analysis by tools

Contents

1 Static Versus Dynamic Techniques

Possible to check products *before* testing is executed

dynamic = execution, test

- Problems: One must wait until something can be executed (code).

static = analysis with tools,
manual review

- Tools: Example spell checker, compiler with warnings, static analyzers
- Review: Formal or informal. Minimum: Self check (”desk test”)
- Problems: Static techniques do not find all faults, some can only be found by dynamic testing.

Any document can be statically analyzed!

Objective and value

Static techniques find the defect, not just a symptom (failure)!

Cheaper than dynamic testing: Finds defects earlier.

Less defects in the product make dynamic testing cheaper, faster and easier.

BUT: Some defects may not be important enough.

Why Reviews?

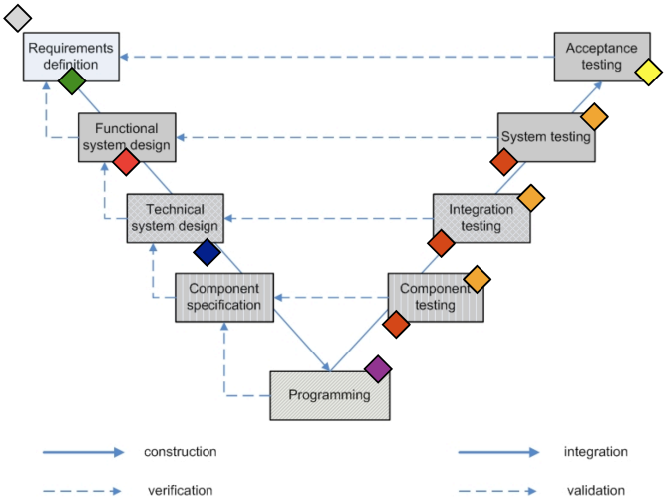
- Review use the most important testing tool there is: People's brain.
- Find defects
- Find forgotten things
- Reduce development and maintenance time and cost
- Improve software quality and reliability
- Reduce cost of testing
- Reduce dependence on testing
- Reduce risk
- Training and learning effect on participants (better common understanding of the system)

Some Faults Reviews Will Find

- Deviations from standards
- Bad design
- Awkward solutions
- Bad maintainability
- Unnecessary complexity
- Faults and mistakes in specifications and interfaces

Reference: Reviews in the V-Model

- Contract review
- Test plan review
- Architecture review
- Detail design review
- Code review
- Test readiness review
- Test exit review
- Acceptance review



Pre-review (not for exam)

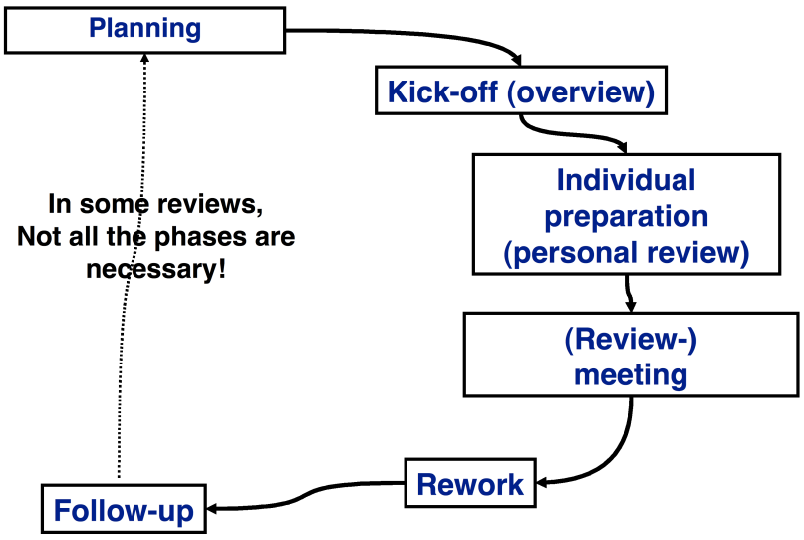
Can the material be reviewed at all?
You meet many excuses.
Very short process with interesting results.

2 Review Process

Formal - documented
Informal - only to give feedback

The other dimension: Systematic / unsystematic
Formality depends on
- Risk
- Requirements to formality
- Requirements to traceability
- Time, resources
- Infrastructure
- Maturity in the organization and development process
- Objective for review (find faults, discussion and consensus, ...)

2.1 Phases (for formal reviews)



Planning

Select participants (- qualified!)
Distribute Roles (see later)
Define start- and exit criteria (for inspection)
What to look for
Plan room
Plan time
Find tools for assistance and help
Necessary background documents
Distribute documents (early enough)

Kick-off (overview)

Before: Check start criteria (f. ex. for inspection)

- Explain the objectives
- Explain the process
- Explain the documents

Start criteria, reasons: As a reviewer you will feel misused if you find too many trivial problems like spelling, grammar, layout or anything else that should have been found by author self-check or tool-supported analysis. The same is true when explicit requirements are left out.

Individual Preparation

Everyone by him/herself, BEFORE the meeting
Use enough time (important)!
Note possible faults
Note questions
Note remarks

Most important part of ALL types of reviews

Examination/evaluation/recording of results (Review meeting):

Logging of faults and comments (worst first!)
Restrictions on free discussion
Decision on repair
Decision on acceptance
Documentation of results, protocol

Max 2 hours!
Max 7 people!
May not be physical meeting (electronic means)

Why A Meeting?

You find a problem->

- Maybe the same type of problem is in other places?
- Maybe there is a deeper cause?
- I did not know this might be a problem…
- Maybe an assuption behind was wrong?

Synergy!

Repair / Rework

Author corrects faults

Other remarks are followed up (comment or correction)

Follow-up

- Check that faults are corrected
- Collect data about the review (size, time, faults found)
- Check exit criteria (f. ex. for inspection)

2.2 Roles and Responsibility

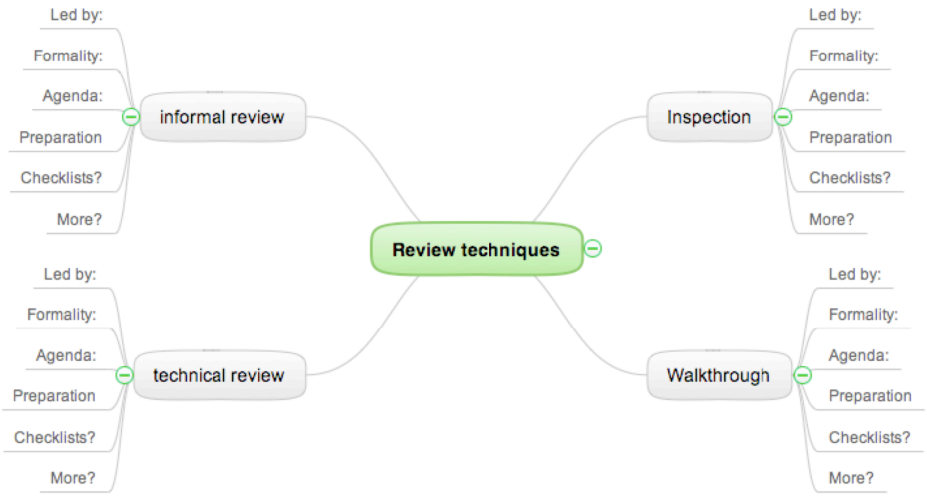
- **Management** - Decision about reviews, allocates time and people, follows up.
- **Moderator** - Leads the review, plans, leads the meeting, collects data, follows up.
- **Author** - Made the document, answers questions. Do not manipulate the review! Repair defects.
- **Reviewer** - ”Expert” in the domain area. Different qualifications. Reviews the document. Typically a colleague of the author (”peer review”). Reviewers should represent different perspectives and roles!
- **Scribe** (recorder) - Notes during the meeting: Faults and other points to follow up.

One person can have several roles!
Best to use check lists!

2.3 Review types

Informal review
Walkthrough
Technical review
Inspection

Questions to remember for your learning (repeat at home)



Informal review

No formal process

No/little documentation that the review was held and how

Very dependent on participants motivation

Most important use: Rapid feedback

Variant: Pair wise work, buddy system

Ref about how to do tool assisted code reviews (May 2011):
http://swreflections.blogspot.com/2011/05/not-doing-code-reviews-whats-your.html

Walkthrough

Formal or informal process

Led by the author

Possible with individual preparation before the meeting, reporting, use of secretary/scribe

Author presents the document, the others comment (evaluate, find problems, check conformance)

Alternative implementations may be considered

Meeting often longer than in other reviews

Use for: Finding faults, teaching, common understanding

The author self usually finds most faults.

Technical Review

From very informal to very formal - Rather formal than informal

Best if led by trained moderator
Individual preparation
May use check lists
Presenter is NOT the author (author does not necessarily participate)
Comments and issues sent to moderator before review meeting
Usually a review report
Participants are colleagues or technical experts
Managers may participate (if they have something to contribute), otherwise "peer review" - persons at the same organisational level

Used for: Discussion/evaluation of alternatives, solve problems, check against specification or standard, find faults

Inspection

Formal process
Led by a moderator (who should be trained)
Formal assignment of roles
Start- and exit criteria
Individual preparation
Use of check lists (see note)
Data collection
Formal follow-up
Infrastructure: Optional use of inspection data to improve the inspection process

Used for: Finding faults, measure document quality

How to choose the review type

No general rules.

It depends on
- Risk
- Time pressure
- Available personnel
- Document type
- Support by management
- Etc.

Several review types may be blended in practice.

2.4 Success Factors

A clear goal
The right participants
Support from management
Participants have time (preparation)
Well led (psychological aspects - comments are important! Not a waste of time)
Right review type
Check lists / Roles
Training, culture, Infrastructure

Reviews used for learning and process improvement

Think About

Can you have several reviews on one document?
- Yes, for example first on the draft, then the final document, or reviews about different topics.

Maximum number of people in a review: 7 (?)

What if we need more than 7? - People do not take responsibility. They find the same faults. Difficult to get all into the meeting.

What if few people are used? - Difficult to cover all aspects.

Length of meeting: max. 2 hours. But several meetings possible.

What if the organization is distributed? - Video conference,phone conference, net meeting.

Should managers participate? - Are people scared? Self-censoring? What about management documents like strategies, standards, rules, processes, plans?

Psychology: Hidden agendas - You do not want to look like an idiot, …

How to improve the inspection process using review data?

Questions?

3 Static Analysis

Analysis of documents, models, code

Without executing the code!

Even on generated documents (XML, HTML) (www.netmechanics.com, www.w3c.org)

What Static Code Analysis Can Find

And more!

Variables not initialized
Variables never used
Code impossible to execute (dead)
Possible endless loops
Deviation from standards
Portability problems
Security holes
Interfaces do not match
Use of elements outside of tables
Complexity analysis
Syntax errors in code and models
Memory leaks
Pointer errors
Ineffective programming solutions

Data flow problems

Control flow problems

Example

```
function wrongcode (int a, b; float c() );
    if (b > a)
    then
        x := c(i);
    else
        while (i < 100) do
                i++; a:= c(i-1);
                ...
                i:=0;
        end while;
    end if;
end;
```

What is wrong here?

Example: Ariane 5 Crash

See http://www.around.com/ariane.html

Why: 64 bit value sent to 16 bit variable.

Complexity and static analysis

Complexity can be quantitatively measured.
Tools do this.

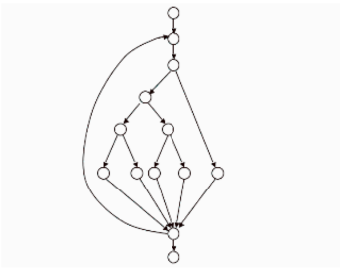
High complexity -> more defects -> more risk.

Analysis can be used for risk management.

Complexity example: Cyclomatic Number (not for exam)

Defined for graphs: #arrows - #boxes + 2 * (# of graphs)
Meaning = Possible paths through the graph.
Good alarm: If more than 10 -> normally too many defects.
In the graph: 17-13+2*1 = 6
Measure with tools!

Many other measures!



Rule of thumb: Number of ”islands” +1.
If number greater than 10, then have a look at the code!

Experiences With Static Analysis

What about your own spell checking?

Example:

I have spelling checker, it came with my PC. It plainly marks four my revue, mistakes eye cannot sea. I've run this poem threw it, I'm sure you're pleased to no. Its letter perfect in its weigh, my checker tolled me saw. - Sauce unknown

Bruel&Kjær, ESSI project:

- Static analysis per fault: 1.6 hours work. (most of this was checking false warnings)
- Faults found in testing: 9 hours work.

The Value of Static Analysis

Finds faults early.
Finds faults, not symptoms.
Can give warnings about too high complexity.
Finds faults which tend to survive (dynamic) testing.
Better maintainability.
Faults can be prevented.
Finds even faults and inconsistencies in software models and interfaces.

More: Gartner Group report no. G00158218 (from 2008)

Good against "idiot errors"

Think about (after the seminar)

- What can your compiler analyze if you switch it to a higher ”warning level”?
- Check freeware and open source tools!
- You must configure the tool: Like with spell checking.
- Global Static Analysis is interesting, but requires large machine resources and special tools (cross-platform, cross-language).
- Special analyzers available (portability, internationalization).
- How to motivate use / what leads to de-motivation?

Questions?

References and Extra Information

- Gartner Group report no. G00158218 (from 2008)
- A.Frank Ackermann, Lynne S. Buchwald, Frank A Lewski, "Software Inspections: An Effective Verification Process", IEEE Software, May 1989
- IEEE Standard 1028-2008: Standard for Software Reviews and Audits, IEEE Computer Society.
- Bisant, D.A., Lyle, J.R., "A Two-Person Inspection Method to Improve Programming Productivity", IEEE Transactions on Software Engineering, October 1989.
- Ebenau, Robert G. and Strauss, Susan H., Software Inspection Process, McGraw Hill, 1994
- Fagan, M. 1976. Design and code inspections to reduce errors in program development. IBM Systems Journal 15, no. 3: 182-211. (Reprinted in IBM Systems Journal 38, no. 2: 259-287 or see www.almaden.ibm.com/journal).
- **Freedman, D. P., Weinberg, G. M.: Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products. 3rd ed., Dorset House Publishing Company, Inc., 1990.**
- Gilb, T., Planning to Get the Most out of Inspection, Software Quality Professional, March 2000
- Gilb, T., and D. Graham. 1993. Software inspections. London: Addison-Wesley Longman.
- Porter, A., Votta, L., "What Makes Inspections Work", IEEE Software, Nov 1997, pg. 99-102.
- Schaefer, H., "Fast software development needs fast reviews", Conference on Views on Software Development in the New Millennium, University of Iceland, August 2000. www.espice.hi.is/rvk
- Søren Skogstad Nielsen, "Inspektion i praksis", Teknologic Institutt, Automatiseringsteknikk, Postboks 141, DK-2630 Taastrup
- Tackett, B. D., Van Doren, Process Control for Error-Free Software, A Software Success Story, IEEE Software 3/1999
  - The paper describes the successful development of an embedded system for the US Air Force using an incremental approach with a lot of weight on "design a little" - "test a little". A key point in the success was the review process of small increments.
- www.codestriker.souceforge.net: Jason Remillard, Source Code Review Systems, IEEE Software Magazine 1/2005 pp 74 ff.