3.1 Static Techniques and the Test Process (K2)	*15 minutes*

Terms
Dynamic testing, static testing

Background
Unlike dynamic testing, which requires the execution of software, static testing techniques rely on the manual examination (reviews) and automated analysis (static analysis) of the code or other project documentation without the execution of the code.

Reviews are a way of testing software work products (including code) and can be performed well before dynamic test execution. Defects detected during reviews early in the life cycle (e.g., defects found in requirements) are often much cheaper to remove than those detected by running tests on the executing code.

A review could be done entirely as a manual activity, but there is also tool support. The main manual activity is to examine a work product and make comments about it. Any software work product can be reviewed, including requirements specifications, design specifications, code, test plans, test specifications, test cases, test scripts, user guides or web pages.

Benefits of reviews include early defect detection and correction, development productivity improvements, reduced development timescales, reduced testing cost and time, lifetime cost reductions, fewer defects and improved communication. Reviews can find omissions, for example, in requirements, which are unlikely to be found in dynamic testing.

Reviews, static analysis and dynamic testing have the same objective – identifying defects. They are complementary; the different techniques can find different types of defects effectively and efficiently. Compared to dynamic testing, static techniques find causes of failures (defects) rather than the failures themselves.

Typical defects that are easier to find in reviews than in dynamic testing include: deviations from standards, requirement defects, design defects, insufficient maintainability and incorrect interface specifications.

3.2 Review Process (K2) — *25 minutes*

Terms
Entry criteria, formal review, informal review, inspection, metric, moderator, peer review, reviewer, scribe, technical review, walkthrough

Background
The different types of reviews vary from informal, characterized by no written instructions for reviewers, to systematic, characterized by team participation, documented results of the review, and documented procedures for conducting the review. The formality of a review process is related to factors such as the maturity of the development process, any legal or regulatory requirements or the need for an audit trail.

The way a review is carried out depends on the agreed objectives of the review (e.g., find defects, gain understanding, educate testers and new team members, or discussion and decision by consensus).

3.2.1 Activities of a Formal Review (K1)

A typical formal review has the following main activities:

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)
2. Kick-off
   - Distributing documents
   - Explaining the objectives, process and documents to the participants
3. Individual preparation
   - Preparing for the review meeting by reviewing the document(s)
   - Noting potential defects, questions and comments
4. Examination/evaluation/recording of results (review meeting)
   - 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
5. Rework
   - Fixing defects found (typically done by the author)
   - Recording updated status of defects (in formal reviews)
6. Follow-up
   - Checking that defects have been addressed
   - Gathering metrics
   - Checking on exit criteria (for more formal review types)

3.2.2 Roles and Responsibilities (K1)

A typical formal review will include the roles below:
- Manager: decides on the execution of reviews, allocates time in project schedules and determines if the review objectives have been met.

- Moderator: the person who leads the review of the document or set of documents, including planning the review, running the meeting, and following-up after the meeting. If necessary, the moderator may mediate between the various points of view and is often the person upon whom the success of the review rests.
- Author: the writer or person with chief responsibility for the document(s) to be reviewed.
- Reviewers: individuals with a specific technical or business background (also called checkers or inspectors) 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, and should take part in any review meetings.
- Scribe (or recorder): documents all the issues, problems and open points that were identified during the meeting.

Looking at software products or related work products from different perspectives and using checklists can make reviews more effective and efficient. For example, a checklist based on various perspectives such as user, maintainer, tester or operations, or a checklist of typical requirements problems may help to uncover previously undetected issues.

3.2.3 Types of Reviews (K2)

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. For example, an informal review may be carried out before a technical review, or an inspection may be carried out on a requirements specification before a walkthrough with customers. The main characteristics, options and purposes of common review types are:

Informal Review

- No formal process
- May take the form of pair programming or 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

Walkthrough

- 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
- Optional scribe (who is not the author)
- May vary in practice from quite informal to very formal
- Main purposes: learning, gaining understanding, finding defects

Technical Review

- Documented, defined defect-detection process that includes peers and technical experts with optional management participation
- May be performed as a peer review without management participation
- Ideally led by trained moderator (not the author)
- Pre-meeting preparation by reviewers
- Optional use of checklists
- Preparation of a review report which includes the list of findings, the verdict whether the software product meets its requirements and, where appropriate, recommendations related to findings
- May vary in practice from quite informal to very formal
- Main purposes: discussing, making decisions, evaluating alternatives, finding defects, solving technical problems and checking conformance to specifications, plans, regulations, and standards

Inspection
- 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 of the software product
- Pre-meeting preparation
- Inspection report including list of findings
- Formal follow-up process (with optional process improvement components)
- Optional reader
- Main purpose: finding defects

Walkthroughs, technical reviews and inspections can be performed within a peer group, i.e., colleagues at the same organizational level. This type of review is called a “peer review”.

3.2.4 Success Factors for Reviews (K2)

Success factors for reviews include:
- Each review has clear predefined objectives
- The right people for the review objectives are involved
- Testers are valued reviewers who contribute to the review and also learn about the product which enables them to prepare tests earlier
- 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)
- The review is conducted in an atmosphere of trust; the outcome will not be used for the evaluation of the participants
- Review techniques are applied that are suitable to achieve the objectives and to the type and level of software work products and reviewers
- Checklists or roles are used if appropriate to increase effectiveness of defect identification
- Training is given in review techniques, especially the more formal techniques such as inspection
- Management supports a good review process (e.g., by incorporating adequate time for review activities in project schedules)
- There is an emphasis on learning and process improvement

3.3 Static Analysis by Tools (K2) — *20 minutes*

Terms
Compiler, complexity, control flow, data flow, static analysis

Background
The objective of static analysis is to find defects in software source code and software models. Static analysis is performed without actually executing the software being examined by the tool; dynamic testing does execute the software code. Static analysis can locate defects that are hard to find in dynamic testing. As with reviews, static analysis finds defects rather than failures. Static analysis tools analyze program code (e.g., control flow and data flow), as well as generated output such as HTML and XML.

The value of static analysis is:
- 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

Typical defects discovered by static analysis tools include:
- 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)
- Overly complicated constructs
- Programming standards violations
- Security vulnerabilities
- Syntax violations of code and software models

Static analysis tools are typically used by developers (checking against predefined rules or programming standards) before and during component and integration testing or when checking-in code to configuration management tools, and by designers during software modeling. Static analysis tools may produce a large number of warning messages, which need to be well-managed to allow the most effective use of the tool.

Compilers may offer some support for static analysis, including the calculation of metrics.

References
3.2 IEEE 1028
3.2.2 Gilb, 1993, van Veenendaal, 2004
3.2.4 Gilb, 1993, IEEE 1028
3.3 van Veenendaal, 2004