Reviews

1. Introduction
2. The Principles of Reviews
3. Types of Reviews
4. Phases of Formal Reviews
5. Introducing Reviews
6. Success Factors for Reviews
7. Review Techniques

Introduction

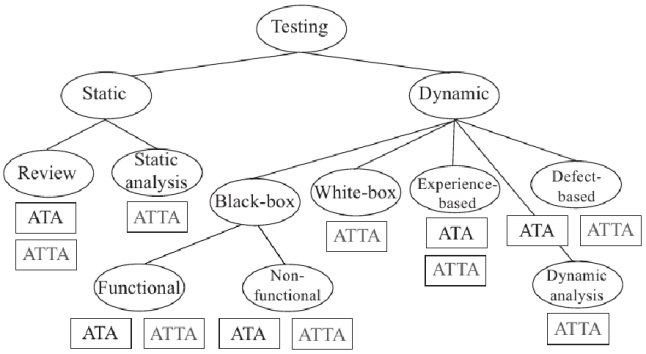

Figure 6–1 *Advanced syllabus testing techniques*

- A review is a type of static test.
- The object being reviewed is not executed or run during the review.
- Good testers make excellent reviewers.
- Good testers have curious minds and a willingness to ask skeptical questions
- Being a group activity, all review participants must commit to well-conducted reviews.
- One negative, confrontational, or belligerent participant can damage the entire process profoundly.

1. Overview
    - Purpose, scope, conformance, organization, application
2. References
3. Definitions
4. Management reviews
    - Responsibilities, inputs/outputs, entry/exit criteria, procedures
5. Technical reviews
    - Responsibilities, inputs/outputs, entry/exit criteria, procedures
6. Inspections
    - Responsibilities, inputs/outputs, entry/exit criteria, procedures, data collection, process improvement
7. Walk-throughs
    - Responsibilities, inputs/outputs, entry/exit criteria, procedures, data collection, process improvement
8. Audits
    - Responsibilities, inputs/outputs, entry/exit criteria, procedures

Figure 6–2 *IEEE 1028 standard for software reviews*

- The document requires some non-trivial changes but not a re-review.
- The most costly outcome—in terms of both effort and schedule time—is that the document requires extensive changes and a re-review.

Informal review:

- There are no defined rules, no defined roles, no defined responsibilities, so you can approach these however you please.

During a formal review, there are some essential roles and responsibilities::

- **The manager.** The manager allocates resources, schedules reviews, and the like. However, the manager might not be allowed to attend based on the review type.
- **The moderator or leader:** This is the chair of the review meeting.
- **The author:** This is the person who wrote the item under review. A review meeting, done properly, should not be a sad or humiliating experience for the author.
- **The reviewers:** These are the people who examine the item under review, possibly finding defects in it. Reviewers can play specialized roles based on their expertise or based on some type of defect they should target.
- **The scribe or secretary or recorder**: This is the person who writes down the findings.

Types of Reviews

- **Informal review**:
  - At the lowest level of formality
  - This can be as simple as two people, the author and a colleague, discussing a design document over the phone.
- **Technical reviews** are more formalized, but still not highly formal.
- **Walk-throughs**
  - are reviews where the author is the moderator and the item under review itself is the agenda.
  - That is, the author leads the review, and in the review, the reviewers go section by section through the item under review.
- **Inspections**
  - are the most formalized reviews.
  - The roles are well defined.
  - Managers may not attend.
  - The author may be neither moderator nor secretary.
  - A reader is typically involved.
- **Management Review:**
  - Common purposes of management reviews are to monitor progress, assess status, and make decisions about some project, system, ongoing activity, or process.

- **Contractual review:**
  - Corresponds to some sort of project milestone, often one linked to payment or continuation of a contract.
  - It could also be a management review for a safety-critical or safety-related system.
  - The review could involve managers, vendors, customers, and technical staff.
  - When a project is going well, these are routine.
  - When a project starts to go poorly, particularly if there are multiple vendors involved, expect massive amounts of time and energy to be spent by each vendor trying to obscure who is responsible for the problems.
- **Requirements reviews:**
  - A requirements review can be informal, it can be a walk-through, it can be a technical review, or it can be an inspection.
  - The scope of a requirements review should be whatever it needs to be.
  - If we are building a safety-critical system, a review or part of a review should consider safety. If we are building a high-availability system, a review or part of a review should consider availability and reliability.
  - For all systems, requirements and the requirements reviews should address both functional and nonfunctional requirements.
  - We can include in a requirements review acceptance criteria and test conditions.
- **Design reviews:**
  - A design review can range from informal to technical reviews and inspections.
  - It can involve technical staff and customers and stakeholders, though you'd expect that as the level of detail becomes more intense, the participants would become more technical.
  - In some organizations, there is a concept of preliminary design review and a critical design review.
  - The preliminary design review is a technical review (also a peer review due to the attendees) where technical peers propose the initial approach to deal with technical issues related to system or test designs.
  - The critical design review covers the proposed design solutions, which can include test cases and procedures in some cases.
- **Operational readiness review, acceptance review, or qualification review**
  - The operational readiness review, acceptance review, or qualification review is a combination of technical and managerial review.
  - This is sort of a final safety net.
  - We want to review all the data to make a final decision on readiness for production.

- Checking the documents improves the moderator ability to lead the meeting because it ensures the better understanding.
- To improve the effectiveness of the review, different roles are assigned to each of the participants.
- These roles help the reviewers focus on particular types of defects during checking.
- This reduces the chance of different reviewers finding the same defects.
- The moderator assigns the roles to the reviewers.

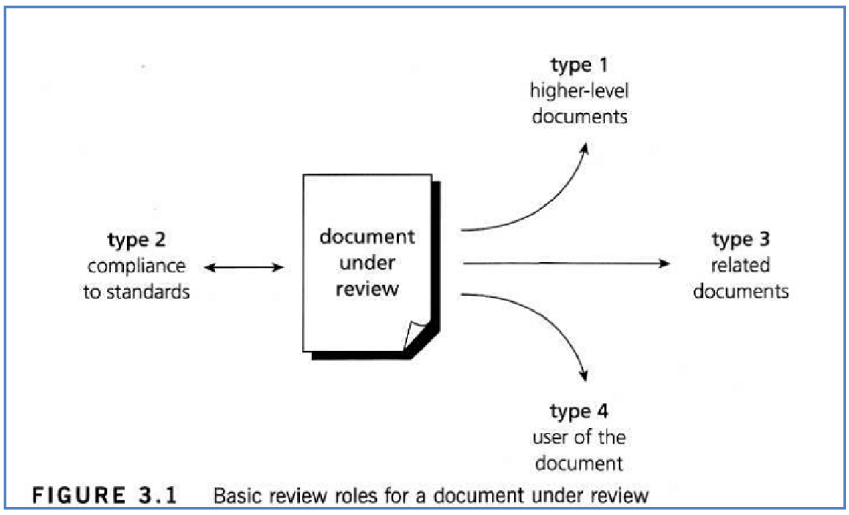

FIGURE 3.1 Basic review roles for a document under review

- Within reviews the following focuses can be identified:
  - focus on higher-level documents, e.g. does the design comply to the requirements;
  - focus on standards, e.g. internal consistency, clarity, naming conventions, templates;
  - focus on related documents at the same level, e.g. interfaces between software functions;
  - focus on usage, e.g. for testability or maintainability.

- **Kickoff Phase**
  - It is an optional step in a review procedure.
  - The goal of this step is to give a short introduction on the objectives of the review and the documents to everyone in the meeting.
  - The relationships between the document under review and the other documents are also explained, especially if the numbers of related documents are high.
  - At customer sites, we have measured results up to 70% more major defects found per page as a result of performing a kick-off
  - Role assignments, checking rate, the pages to be checked, process changes and possible other questions are also discussed during this meeting.

        - As chairman of the discussion meeting, the moderator takes care of the people issues and prevents discussion from getting too personal and calls for a break to cool down the heated discussion.
        - The outcome of the discussions is documented for the future reference.
    - **Decision phase**:
        - At the end of the meeting a decision on the document under review has to be made by the participants, sometimes based on formal exit criteria.
        - Exit criteria are the average number of critical and/or major defects found per page (for example no more than three critical/major defects per page).
        - If the number of defects found per page is more than a certain level then the document must be reviewed again, after it has been reworked.

- **Rework Phase:**
    - In this step if the number of defects found per page exceeds the certain level then the document has to be reworked.
    - Not every defect that is found leads to rework. It is the author's responsibility to judge whether the defect has to be fixed.
    - If nothing can be done about an issue then at least it should be indicated that the author has considered the issue.
    - Changes that are made to the document should be easy to identify during follow-up. Therefore the author has to indicate where changes are made

- **Follow-up Phase:**
    - In this step the moderator check to make sure that the author has taken action on all known defects.
    - If it is decided that all participants will check the updated documents then the moderator takes care of the distribution and collects the feedback.
    - It is the responsibility of the moderator to ensure that the information is correct and stored for future analysis.

Introducing Reviews

- The following steps are useful in successfully introducing reviews:
    - **Secure management support:** Reviews are not expensive from a budget point of view, as test automation is, but they do require a time commitment, especially when time is tight.

- Checklists are helpful for reviews.
  - It's too easy to forget important areas without them.
  - Have some checklists.
  - They should address common defects based on what you find.
  - Also, have different checklists for different kinds of documents, such as requirements, use cases, and designs.
  - Finally, have different checklists for different review processes.
- The appropriate level of formality varies. So be ready to use more than one type of review.
- All documents are vitally important.
  - If a document is involved in making an important decision, such as signing a contract, be sure to inspect the proposal, contract, or high-level requirements specification first.
  - If a major expenditure is being contemplated, have a management review to authorize it.
  - For large documents, you can use a sampling of a limited subset to estimate the number of defects in the entire document.
  - This can help to determine if a re-review is needed.
  - Keep in mind that this sampling approach won't work for a document cleanup or edit. Watch out for distractions.
  - It's easy to find a bunch of minor format, spelling, and grammar errors.
  - Focus on finding the most important defects, based on content not format.
  - Finally, continuously improve the review process.

- **Organizational factor:**
  - Make sure managers will plan and estimate for adequate time, especially under deadline pressures.
  - Be careful with the metrics. For one thing, remember that some reviews will find many defects per person-hour invested, while others won't.
  - Choose appropriate model for predicting defect density.
  - Never ever let review defect metrics be used for individual performance evaluations.
  - Make sure to allow time for rework of defects.
  - Make sure the process involves the right participants.
  - Ensure that the participants participate.
  - Ensure proper time for preparation is given.
  - Another part is to draw less-vocal participants into the meeting. Just because someone doesn't have a forceful personality doesn't mean they don't have good ideas.

- The network is homogenous.

- **Marick's Code Review Checklist**
  - Technical test analysts are likely to be invited to code reviews.
  - So, let's go through Brian Marick's code review checklist, which he calls a "question catalog." This catalog has several categories of questions that developers should ask themselves when going through their code.
  - These questions are useful for many procedural and object-oriented programming languages, though in some cases certain questions might not apply.
  - For variable declarations, Marick says we should ask the following questions:
    - Are the literal values correct? How do we know?
    - Has every single variable been set to a known value before first use? When the code changes, it is easy to miss changing these.
    - Have we picked the right data type for the need? Can the value ever go negative?
  - For each data item and operations on data items, Marick says we should ask the following questions:
    - Are all strings NULL terminated? If we have shortened or lengthened a string, or processed it in any way, did the final byte get changed?
    - Did we check every assignment to a buffer for length?
    - When using bitfields, are our manipulations (shifts, rotates, etc.) going to be portable to other architectures and endian schemes?
    - Does every sizeof() function call actually go to the object we meant it to?
  - For every allocation, deallocation, and reallocation of memory, Marick says we should ask the following questions:
    - Is the amount of memory sufficient to the purpose without being wasteful?
    - How will the memory be initialized?
    - Are all fields being initialized correctly if it is a complex data structure?
    - Is the memory freed correctly after use?
    - Do we ever have side effects from static storage in functions or methods?
    - After reallocating memory, do we still have any pointers to the old memory location?
    - Is there any chance that the memory might be freed multiple times?
    - After deallocation, are there still pointers to the memory?
    - Are we mistakenly freeing data we don't mean to?

- In terms of coding standards, here are some additional questions to ask (assuming you are not enforcing coding standards via static analysis):
  - Do all of the code changes meet our standards and guidelines? If not, why not?
  - Are all data values to be passed parameterized correctly? In terms of design changes, here are the questions to ask:
  - Do you understand the design changes and the reasons they were made?
  - Does the actual implementation match the designs?
- Here are the maintainability questions to ask:
  - Are there enough comments? Are they correct and sufficient?
  - Are all variables documented with enough information to understand why they were chosen?
- Finally, here are the documentation questions included in the OpenLaszlo checklist:
  - Are all command-line arguments documented?
  - Are all environmental variables needed to run the system defined and documented?
  - Has all user-facing functionality been documented in the user manual and help file?
  - Does the implementation match the documentation?

****