Software Inspections and Walkthroughs

Software Inspections

- “Software Inspections are a disciplined engineering practice for detecting and correcting defects in software artifacts, and preventing their leakage into field operations.” Don O'Neill, Don O' Neill Consulting for SEI

Software Walkthroughs

- a form of software peer review "in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems" (IEEE Std. 1028-1997, *IEEE Standard for Software Reviews*, clause 38.

What's the difference?

- An inspection is a more formal process than a walkthrough used to collect metrics or statistics about the software process
- Walkthrough is a more informal version of an inspection

Why inspect software?

- Routine production of reliable software within budget and on schedule continues to elude most development organizations.
- While efforts are ongoing to make development an industrial process, much of the work is still done by “intellectual artisans”
- Artisan’s work is inherently difficult to bond and can not be specified precisely.
- Inspections are a method to reduce variability and tighten process control.

History of Inspections

- M.E. Fagan of IBM first defined inspections in 1976.
- E. Yourdon was among the first to publish a book on inspections in 1978.
- IEEE standard covering inspections first appeared in 1988.

What do inspections cover?

- Inspections and walkthroughs are primarily intended to discover defects in software artifacts.
- This is a static analysis technique of software testing.
- In addition, inspections address three major tasks of process management: planning, measurement, control.

Inspection metrics

- Inspections are used to collect quantitative quality data at defined points in the development process.
- This can be used to give feedback to the developers, feed-forward to future development, and feed-into future steps of process.
- Can also provide data on effectiveness of inspection techniques.

What can be inspected?

- Inspections can be held a various points in development process.
- Fagan recommended inspections on:
  - Detailed design
  - Cleanly compiled code
  - Completion of unit test

Who is involved?

- At a minimum a formal inspection includes:
  - Designated moderator
  - Author of the work
  - At least one peer inspector
- Walkthroughs generally do not include designated moderator and are often led by the author of the software.

Steps of inspection

- Planning
- Overview
- Preparation
- Meeting
- Rework
- Follow-up

Inspection Roles

- Author – developer of work product
- Moderator – an inspector responsible for organizing and reporting on inspection
- Reader – an inspector who guies the examination of the product
- Recorder – an inspector who enters all the defects found on the defect list
- Inspector – Member of inspection team. Often chosen to represent specific role- designer, tester, technical writer, SQA, etc

Conclusions

- Inspections have been proven an efficient and effective method for improving software quality
- In conjunction with testing, audits and formal verification a successful, quality product can be produced

Detailed Design

Detailed Design

In previous chapters we discussed two different approaches for system design. In system design we concentrate on the modules in a system and how they interact with each other. Once a module is precisely specified, the internal logic that will implement the given specifications can be decided, and is the focus of this chapter. In this chapter we discuss methods for developing and specifying the detailed design of a module. We also discuss the metrics that can be extracted from a detailed design.

Detailed Design

Most design techniques, like structured design, identify the major modules and the major data flow among them. The methods used to specify the system design typically focus on the external interfaces of the modules and cannot be extended to specify the internals. Process design language (PDL) is one way in which the design can be communicated precisely and completely to whatever degree of detail desired by the designer. That is, it can be used to specify the system design and to extend it to include the logic design. PDL is particularly useful when using top-down refinement techniques to design a system or module.

Verification

There are a few techniques available to verify that the detailed design is consistent with the system design. The focus of verification in the detailed design phase is on showing that the detailed design meets the specifications laid down in the system design. Validating that the system as designed is consistent with the requirements of the system is not stressed during detailed design. The three verification methods we consider are design walkthroughs, critical design review, and consistency checkers.

Design Walkthrough

A *design walkthrough* is a manual method of verification. The definition and use of walkthroughs change from organization to organization. Here we describe one walkthrough model. A design walkthrough is done in an informal meeting called by the designer or the leader of the designer's group. The walkthrough group is usually small and contains, along with the designer, the group leader and/or another designer of the group. The designer might just get together with a colleague for the walkthrough or the group leader might require the designer to have the walkthrough with him.

In a walkthrough the designer explains the logic step by step, and the members of the group ask questions, point out possible errors or seek clarification. A beneficial side effect of walkthroughs is that in the process of articulating and explaining the design in detail, the designer himself can uncover some of the errors.

Walkthroughs are essentially a form of peer review. Due to its informal nature, they are usually not as effective as the design review.

Critical Design Review

The purpose of *critical design review* is to ensure that the detailed design satisfies the specifications laid down during system design. The critical design review process is same as the inspections process in which a group of people get together to discuss the design with the aim of revealing design errors or undesirable properties. The review group includes, besides the author of the detailed design, a member of the system design team, the programmer responsible for ultimately coding the module(s) under review, and an independent software quality engineer. While doing design review it should be kept in mind that the aim is to uncover design errors, not try to fix them. Fixing is done later.

Critical Design Review

The use of checklists, as with other reviews, is considered important for the success of the review. The checklist is a means of focusing the discussion or the "search" of errors. Checklists can be used by each member during private study of the design and during the review meeting. For best results, the checklist should be tailored to the project at hand, to uncover project-specific errors. Here we list a few general items that can be used to construct a checklist for a design review [52].

A Sample Checklist

- Does each of the modules in the system design exist in detailed design?
- Are there analyses to demonstrate that the performance requirements can be met?

Critical Design Review

- Are all the assumptions explicitly stated, and are they acceptable?
- Are all relevant aspects of system design reflected in detailed design?
- Have the exceptional conditions been handled?
- Are all the data formats consistent with the system design?
- Is the design structured, and does it conform to local standards?
- Are the sizes of data structures estimated? Are provisions made to guard against overflow?
- Is each statement specified in natural language easily codable?
- Are the loop termination conditions properly specified?
- Are the conditions in the loops OK?

Consistency Checkers

Design reviews and walkthroughs are manual processes; the people involved in the review and walkthrough determine the errors in the design. If the design is specified in PDL or some other formally defined design language, it is possible to detect some design defects by using consistency checkers.

Consistency checkers are essentially compilers that take as input the design specified in a design language (PDL in our case). Clearly, they cannot produce executable code because the inner syntax of PDL allows natural language and many activities are specified in the natural language. However, the module interface specifications (which belong to outer syntax) are specified formally. A consistency checker can ensure that any modules invoked or used by a given module actually exist in the design and that the interface used by the caller is consistent with the interface definition of the called module. It can also check if the used global data items are indeed defined globally in the design.