Unit 2. Software Inspection

1. Introduction
2. Types of Software Reviews
3. Software Inspection
4. Inspection Checks
5. Inspection Metrics

Reading: TB-Chapter 3 (Section 3.2)

1. Introduction

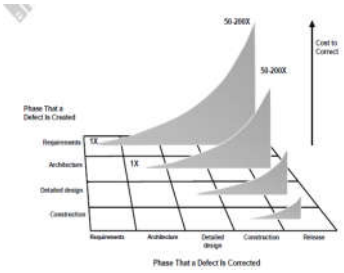
-Quality control activities determine whether a product conforms to its ***requirements***, ***specifications***, or ***pertinent standards***.

•Nonconformance constitute defects.

-Performing quality control within the development process is more efficient than looking for errors after the product has been completed.

-Software review is a ***verification*** method that is used to improve work products before they are completed.

•Consist of having people other than the author of a work product examine that product to find defects and identify improvement opportunities.



-Software review is essential for uncovering a host of defects that other quality control techniques such as testing or static analysis are not adequate for or would miss.

-These include, for instance:

•Product compliance with standards

•Detecting logic errors in implementation

•Spotting erroneous, ambiguous, or missing requirements

•Judging clarity or maintainability of the code

2. Types of Software Reviews

-There are various types of review based on the degree of formality:

•**Buddy checking**:

-Consist of having a person other than the author ***informally*** review a document or work.

-In general this doesn't involve the use of checklists to guide the process and as such is not repeatable.

•**Walkthrough**:

-Involve the author of an artifact presenting it to an audience of their peers, and receiving comments and feedbacks.

-Usually, such process involves limited documentation of the process and the issues uncovered, which makes defect tracking difficult.

Types of Software Reviews (ctd.)

•**Review by circulation**:

-Consist of circulating an artifact among peers for comments; operates like a walkthrough but without holding a meeting.

-Not holding a meeting avoids potential arguments over issues, but also the benefits of discussion.

•**Inspection**: ***formal*** and ***managed*** peer review process with the following characteristics:

-The process is carried out by a review team with clearly defined roles.

-The process follows an unambiguous set of criteria for each type of artifacts.

-Specific data are collected during the process.

-The process is driven by quantitative goals such as process and quality improvements.

3. Software Inspection

Overview

-Introduced by Michael Fagan in the 1970s, as an elaborated process for systematic review of software artifacts.

-Has been used on a wide variety of software artifacts: code as well as all the other documents such as specification, design, and test data.

-Main issues:

•Inspection can be very expensive because of the head counts (4-5 persons) and time required.

•Inspection outcomes depend on the experience and training of the inspectors.

Overview (ctd.)

-Still, inspection can be very effective when applied to a project from start to finish.

-According to data collected from various studies, the benefits of inspection include the following:

- Net increase in productivity in the range of 30-100%;
- Overall project time saving of 10-30%;
- 5 to 10 times reduction in test execution costs and time;
- Reduction in maintenance costs of up to one order of magnitude;
- Improvement in consequent product quality.
- Minimal defect correction backlash at systems integration time.

Overview (ctd.)

-According to many previous studies inspection is more efficient than testing, specifically when comparing inspection of code modules with testing those modules.

- Inspection finds most of the problems testing would find, and does so more efficiently.

-However, some subtle bugs may escape inspection, and could only be detected through testing by executing the software code.

- So, inspection and testing should be considered as complementary processes.

Inspection Participants and Procedure

Participants

-The following key participants are involved in a formal inspection process:

- **Author (or Owner):**
  -the person who wrote the document or work product under review;
  -his role in the inspection consists solely of presenting his work, and helping others understand it, but not of "defending" it.

- **Moderator:**
  -runs the inspection, by rigorously enforcing its purpose, which is to discover deficiencies in the document being inspected.

- **Recorder:**
  -is in charge of recording possible deficiencies uncovered during the inspection process, and along with the moderator, of preparing at the end of the process, an inspection report, which will be used to fix the problems found.

Inspection Participants and Procedure (ctd.)

Participants (ctd.)

- **Reader:**
  -Paraphrases the code or document at an inspection meeting.

- **Inspectors:**
  -are the ones who raise questions, suggest problems, and criticize the document (but are not supposed to "attack" it).
  -may include people from different expertise (e.g., quality assurance, marketing etc.), each bringing a different viewpoint.

Procedure

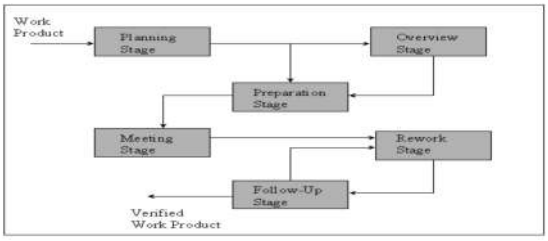
-All the participants are supposed to study the document in advance, and identify issues to raise.

-Inspectors work from checklists of things to look for, lists to which they add as their experience accumulates.

-Formal inspection requires a minimum of four people and is best with seven participants.

- In small organizations, the personnel don't exist to systematically carry it out.
- In such case, if two engineers work well together, they can do useful inspection with no moderator, and one of them acting as the recorder.

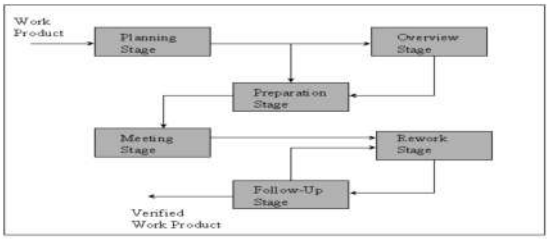
Inspection Process



- **Planning:** Identifies work product to be inspected, determines the size and composition of the inspection team, and sets the inspection schedule.

- **Overview:** Optional phase where team members who are unfamiliar with the work product to be inspected receive orientation.

Inspection Process



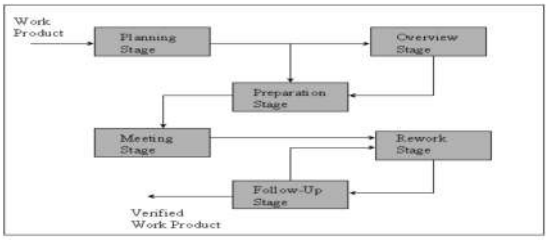
•**Preparation:** Team members inspect the work individually looking for defects in the work product with the guidance of relevant checklist.

-The majority of defects found in inspection processes are found in this stage (about 75%);

-The reviewer should record any issues found, and determine their severity (i.e., ***major*** or ***minor***):
- •***major*** for a defect that will cause an observable product failure or departure from requirements;
- •***minor*** for a defect that will not cause a failure in execution of the product.

-At this stage, it is strictly required that reviewers work individually and do not attempt to find solutions to defects found as this is an improper use of their time.

Inspection Process (ctd.)



•**Inspection Meeting:** Inspection team members meet to discuss possible defects in the work product. The moderator should ensure that all the issues raised by individual reviewers are appropriately logged.

•**Rework:** The work product is revised to conform to requirements and specifications.

•**Follow up:** The rework is verified, final inspection data is collected and summarized, and the inspection is officially closed. At this stage also, the moderator may calculate certain metrics to assess the effectiveness of the inspection and recommend improvement to the inspection process.

-Example of Individual Reviewer Log form (used during preparation stage)

Reviewer: Joe Blogg
Team: DT1
Part level: WCMS
Cycle: 1

Date: 10 March 2003
Owner: M. Bean
Moderator: M. Python

Start Time	10:15
End Time	11:40
Interruptions	25 min
Delta review time	60 min

Location	Defect Description	Severity	Classification
line 5, function swap()	Parameters are passed by value, not by reference. "swap" doesn't correctly swap the numbers, so the sort is not carried out correctly.	major	Function calls
line 12, function max()	Function max() is defined, but never used. No failure apparently, but checklist violation.	minor	Function calls
line 23, function main()	>= should be >. The program only accepts one less than the true maximum number of elements.	major	Comparisons
line 35, function main()	"list" is spelled incorrectly in the message. The program displays incorrect output.	minor	Output format

Summary

Number of major defects: 2
Number of minor defects: 2
Size of reviewed artifacts: 100 LOC

-Example of Log form (used during meeting phase)

Name	Defects		Preparation Data			Estimated yield
	Major	Minor	Size	Time	Rate	
J.B.	2	2	100 Loc	60 min		
R.M	3	5	100 Loc	75 min		
Totals				67.5 min		

No.	Defect Description	Engineers (finding major defects)	
		J.B.	R.M.
line 5, function swap()	Parameters are passed by value, not by reference. "swap" doesn't correctly swap the numbers, so the sort is not carried out correctly.	1	1
line 23, function main()	>= should be >. The program only accepts one less than the true maximum number of elements.	1	
Line 35, function main()	default case missing in switch statement		1
Line 3, function swap	Variable type conversion inadequate		1
Totals		2	3
Unique defects		1	2

Inspection Summary Product size: 100 Loc Meeting Time: 30 min

Total defects for J.B: 2 Total defects for R.M: 3 Common defects: 1

Est. total defects: Total number found: Number left:

Inspection Workload

-The amount of code which can be inspected in a given time depends on the experience of the inspection team, the programming language and the application domain.

-Measurement data collected at IBM, and confirmed by AT&T led to the following observations:

1. About 500 source code statements per hour can be considered during the overview stage.
2. During individual preparation, about 125 source code statements per hour can be examined.
3. From 90 to 125 statements per hour can be inspected during the meeting.

-It was suggested that the maximum time spent on an inspection should be about two hours as the efficiency of the defect detection process falls off after that time.

•Inspection should therefore be a frequent process, carried out on relatively small software components, during program development (≈1-2 hours prep + 1h meeting)

4. Inspection Checks

-The inspection process should always be driven by a checklist of common programming errors.

•Established by discussion with experienced staff and regularly updated as more experience is gained of the inspection process.

•Different checklists should be prepared for different programming languages.

Example of inspection checks

Fault class	Inspection Checks
Data Faults	Are all program variables initialized before their values are used? Have all constant been named? Is there any possibility of buffer overflow? Etc.
Control Faults	For each conditional statement, is the condition correct? Is each loop certain to terminate? Are compound statements correctly bracketed? Etc.
Input/output faults	Are all input variables used? Are all output variables assigned a value before they are output? Can unexpected inputs cause corruption? Etc.
Interface faults	Do all function and method calls have the correct number of parameters? Do formal and actual parameter types match? Are the parameter in right order? If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	If a linked structure is modified, have all links been correctly reassigned? If dynamic storage is used, has space been allocated correctly? Is space explicitly de-allocated after it is no longer required? Etc.
Exception management faults	Have all possible error conditions been taken into account?

Sample Java Code Inspection Checklists

Source: Praktikum Software Engineering 99

1. Variable and Constant Declaration Defects (VC)
1. Are descriptive variable and constant names used in accord with naming conventions?
2. Are there variables with confusingly similar names?
3. Is every variable properly initialized?
4. Could any non-local variables be made local?
5. Are there literal constants that should be named constants?
6. Are there macros that should be constants?
7. Are there variables that should be constants?

2. Function Definition Defects (FD)
8. Are descriptive function names used in accord with naming conventions?
9. Is every function parameter value checked before being used?
10. For every function: Does it return the correct value at every function return point?

3. Class Definition Defects (CD)
11. Does each class have an appropriate constructor and destructor?
12. For each member of every class: Could access to the member be further restricted?
13. Do any derived classes have common members that should be in the base class?
14. Can the class inheritance hierarchy be simplified?

Example: *Sample buggy code patterns*

```
//Eclipse .0
//org.eclipse.jdt.internal.ui.compare
//JavaStructureDiffViewer.java, line 31

…
Control c = getControl();
if (c == null && c.isDisposed()) return;
…
```


Example: *Sample buggy code patterns (ctd.)*

```
//The sample code defines an isName() method that takes a String argument and returns true
// if the given string is a valid name. A valid name is defined as two capitalized words
//separated by one or more spaces.

        public class Foo {

          public boolean isName(String s) {
            String names[] = s.split(" ");
            if (names.length != 2) {
              return false;
            }
            return (isCapitalized(names[0]) && isCapitalized(names[1]));
          }

          public boolean testString(String s) {
            if (s == null) return false;
            else return isName(s);
          }
        }
```


Example: *Sample buggy code patterns (ctd.)*

```
…
attribute = parseAttribute(isEmpty, psp, asp);

if (attribute == null) {
        …
        return;
}

value = parseValue(attribute, false, empty, delim);

if(attribute != null) {
        …
}
else {
        av = new AttVal (null, null, null, null, attribute, value);
        Report.attrError(this, this.token, value, Report.BAD_ATTRIBUTE_VALUE);
}
…
```


Example: *Sample buggy code patterns (ctd.)*

```
import java.io.*;

public class ParseFile {

   public String read (String fname)  {
          try{

                  String filename = System.getProperty("user.home")+
                                    System.getProperty("file.separator")+ fname;
                  BufferedReader br = new BufferedReader(new FileReader(filename));
                  String ior = br.readLine();
                  return ior;

                  } catch (Exception e) {

          }
   }

}
```

Example: *Samples buggy code patterns (ctd.)*

```
class Calendar {

        public void printMonth(int m) {
                int month = m;

                switch (month) {
                        case 1: System.out.println("January"); break;
                        case 2: System.out.println("February"); break;
                        case 3: System.out.println("March"); break;
                        case 4: System.out.println("April"); break;
                        case 5: System.out.println("May"); break;
                        case 6: System.out.println("June"); break;
                        case 7: System.out.println("July"); break;
                        case 8: System.out.println("August"); break;
                        case 9: System.out.println("September"); break;
                        case 10: System.out.println("October"); break;
                        case 11: System.out.println("November"); break;
                        case 12: System.out.println("December"); break;

                }
        }
}
```


5. Inspection Metrics

-The data collected during a software process is used to compute a set of metrics that support evaluation and improvement of the process as well as planning and tracking quality.

-The metrics computed during such process should be defined by the requirements of your organization (typically in the quality manual).

•The collection of data and calculation of ***metrics for no reason is a waste of time***.

-Many different metrics can be calculated during an inspection process including the following:

- **The number of major and minor defects found**
- **The number of major defects found to total found.** (If the proportion of minor defects to major defects is too large a moderator may request that the reviewer repeat the review, focusing on major defects, prior to commencing the logging meeting)
- **The size of the artifact (pages, LOC, ...)**
- **The rate of review -** the size of the reviewed artifact divided by time (normally expressed in hours) (e.g., 15 pages /hour).
- **The defect detection rate -** the number of major defects found per review hour.

Total Number of Defects Found and Defects Density

-Total number of defects found is the sum of the total number of defects found by each reviewer, minus the number of common defects found.

•For instance, with 2 reviewers, the metric is computed by

Total Defects Found = A + B - C

Where A and B are the number found by reviewers A and B respectively and C is the number found by both A and B.

-Defect density is the ratio of the number of defects found to the size of the artifact. It is given by

Defect Density = Total Defects Found / Size

Where the size of the artifact is measured in number of pages, loc, or other size measure.

Estimated Total Number of Defects

-The estimated total number of defects is the sum of the total number of defects found and the estimated total number of defects remaining.

•To estimate the total number of defects remaining in an artifact immediately after inspection, we use an approach similar to the population sampling approach used by biologists to estimate the population of a particular ecosystem.

Population Sampling Approach

-Suppose we have a fish farm and we want to estimate the total number of fish we have. We could apply the capture-recapture method:

1. Capture a number of fish, tag them and release them (let this number be S_1).
2. Allow time for the first sample population to redistribute.
3. Capture a second number of fish (let this number be S_2).
4. Count the number of tagged fish in the second population (let this number be S_T).
5. Calculate the proportion of tagged fish in the second population (let this number be T, then $T = S_T / S_2$).
6. We assume that T is representative of the proportion of tagged fish in the total population (S), so $T \times S = S_1$, or for our purposes $S = S_1 / T$.

-Using the population sampling approach to estimate the number of defects remaining leads to the following steps:

Let the number of defects found by one reviewer be the tagged population (A).

1. Assume an even likelihood of finding all defects (even distribution,...)
2. Count the number of defects found by the second reviewer (B).
3. Count the number of defects found by the second reviewer that were also found by the first (C the common defects).
4. Calculate the proportion of common defects in the second reviewer's defects (T=C/B).
5. We assume that T is representative of the proportion of common defects in the total number of defects (EstTot), so $T \times EstTot = A$, or for our purposes **$EstTot = A/T = (A \times B)/C$.**

-So assuming that defects are equally likely to be found in an artifact and each reviewer is equally likely to find every defect, then:

Estimated Total Defects = (A × B)/C

•Note that in practice such assumptions are not always fulfilled, simply because some defects are harder to find than others, and some reviewers are better than others.

Example: compute inspection metrics from the following inspection log form

Name	Defects		Preparation Data			Estimated yield
	Major	Minor	Size	Time	Rate	
J.B.	2	2	100 Loc	60 min		
R.M	3	5	100 Loc	75 min		
Totals				67.5 min		

No.	Defect Description	Engineers (finding major defects)	
		J.B.	R.M.
line 5, function swap()	Parameters are passed by value, not by reference. "swap" doesn't correctly swap the numbers, so the sort is not carried out correctly.	1	1
line 23, function main()	>= should be >. The program only accepts one less than the true maximum number of elements.	1	
Line 35, function main()	default case missing in switch statement		1
Line 3, function swap	Variable type conversion inadequate		1
Totals		2	3
Unique defects		1	2

Inspection Summary Product size: 100 Loc Meeting time: 30 min

Total defects for J.B: 2 Total defects for R.M: 3 Common defects: 1

Est. total defects: 6 Total number found: 4 Number left: 2

Inspection Yield

-Inspection yield refers to the defect removal efficiency (**DRE**) of an inspection process.

-The defect removal effictiveness of a process is given by calculating the percentage of total defects that were found by that process. So the yield of an inspection is given by:

Yield = Total Defects Found / Estimated Total Defects × 100%

Inspection Rate and Defect Detection Rate

-Requires computing the ***inspection time***, which is the sum of each reviewer's review time plus the total person time spent in each meeting.

-The inspection rate is computed by:

Inspection rate = size / total inspection time

Where:
- *size* stands for the size of the artifact in number of pages, loc, or other size measure.
- total inspection time measured in hours

-The defect detection rate is computed by:

Defect finding efficiency = Total defects found / total inspection time

-Example of Log form (used during meeting phase)

Name	Defects		Preparation Data			Estimated yield
	Major	Minor	Size	Time	Rate	
J.B.	2	2	100 Loc	60 min		
R.M	3	5	100 Loc	75 min		
Totals				67.5 min		

No.	Defect Description	Engineers (finding major defects)	
		J.B.	R.M.
line 5, function swap()	Parameters are passed by value, not by reference. "swap" doesn't correctly swap the numbers, so the sort is not carried out correctly.	1	1
line 23, function main()	>= should be >. The program only accepts one less than the true maximum number of elements.	1	
Line 35, function main()	default case missing in switch statement		1
Line 3, function swap	Variable type conversion inadequate		1
Totals		2	3
Unique defects		1	2

Inspection Summary Product size: 100 Loc Meeting Time: 30 min. Total insp. Time: 255min=4.25hrs

Total defects for J.B: 2 Total defects for R.M: 3 Common defects: 1 Insp. Rate: 23.5LOC/hr

Est. total defects: 6 Total number found: 4 Number left: 2 DFE: 0.94 def/hr DRE: 66%

Calculating Inspection Metrics with more than two Reviewers

-If there are more than 2 reviewers, the same approach can be taken to calculate *inspection totals* and *yield* by grouping reviewers into two groups for calculation: group A and group B:

- If there are 3 reviewers, it is often a good idea to choose the person who has the most unique defects to be one group and the other two reviewers to be the other group. For each group if any member of that group has found a defect, then count it for the group.

Example with 3 reviewers:

No.	Defect Description	Engineers (finding major defects)				
		R1	R2	R3	A (R2)	B
		1	1	1	1	1
			1	1	1	1
		1		1		1
		1	1	1	1	1
		1	1	1	1	1
			1	1	1	1
		1	1		1	1
		1				1
			1		1	
			1		1	
		1	1	1	1	1
Totals		7	9	7	9	9
Unique defects		1	2	0		

Inspection Summary **Product size: 10** **Size measure:**

Total defects for A: **Total defects for B:** **Common defects:**

Est. total defects: **Total number found:** **Number left:**