Software Inspections

6

Abstract

This chapter discusses software inspections, which play an important role in building quality into a product. The well-known Fagan inspection process that was developed at IBM in the 1970s is discussed, as well as lighter review and walkthrough methodologies.

Keywords

Informal review · Structured walk-through · Fagan inspection · Gilb inspections · Economic benefits of inspections · Inspection guides · Entry and exit criteria · Automated software inspections

6.1 Introduction

The objective of software inspections is to build quality into the software product, rather than adding quality later. There is clear evidence that the cost of correction of a defect increases the later that it is detected, and it is therefore more cost effective to build quality in rather than adding it later in the development cycle. Software inspections are an effective way of doing this.

There are several approaches to software inspections, and these vary in the formality of the process. An informal review consists of a walk-through of the document or code by an individual other than the author. The meeting usually takes place at the author's desk (or in a meeting room), and the reviewer and author discuss the document or code informally.

There are formal software inspection methodologies such as the well-known *Fagan inspection* methodology [1] and the Gilb methodology [2]. These methodologies include pre-inspection activity, an inspection meeting and post-inspection

G. O'Regan, *Concise Guide to Software Engineering*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-57750-0_6

Fig. 6.1 Michael Fagan

activity. Several inspection roles are typically employed, including an *author* role, an *inspector* role, a *tester* role and a *moderator* role.

The Fagan inspection methodology was developed by Michael Fagan (Fig. 6.1) at IBM in the mid-1970s, and Tom Gilb developed Gilb's approach in the early 1990s. The formality of the software inspection methodology employed is influenced by the impacts of software failure on the customer's business, as a failure may have a major negative impact on the customer. For example, an incorrect one-line change to telecommunications software could lead to failure resulting in a major telecommunications outage and significant disruption to customers.

Further, there may be financial impacts, as the service level agreement details the service level that will be provided, and the compensation given for service disruption. Consequently, a telecommunications company needs to ensure that its software is fit for purpose, and a formal software inspection process tends to be employed to ensure that quality is built in. This means that requirement documents, high-level and detailed design documents and software code are all inspected, and generally inspections are explicitly planned in the project schedule.

Another words, an organization needs to define an inspection process that is appropriate to its business, and it may adopt a rigorous approach such as the Fagan or Gilb methodology, or a less formal process where the impact of failure is less severe. It may not be possible to have all of the participants present in a room, and

participation by conference call or video link may need to be employed. A formal process may not suit some organizations, and a structured walk-through may be the adopted approach.

Software inspections play an important role in building quality into the software, and in ensuring that the quality of the delivered product is good. The quality of the delivered software product is only as good as the quality at the end each phase, and therefore a phase should be exited only when the desired quality has been achieved.

The effectiveness of an inspection is influenced by the expertise of the inspectors, adequate preparation, the speed of the inspection and compliance to the inspection process. The inspection methodology provides guidelines on the inspection and preparation rates for an inspection, and guidelines on the entry and exit criteria for an inspection.

There are typically at least two roles in the inspection methodology. These include the *author* role and the *inspector* role. The *moderator, tester* and the *reader* roles may also be present in the methodology.

The next section describes the benefits of software inspections, and this is followed by a discussion of a simple review methodology where the reviewers send comments directly to the author. Then, a structured walk-through and a semi-formal review process are described, and finally the Fagan inspection process is described in detail.

6.2 Economic Benefits of Software Inspections

There is clear evidence that a software inspection program provides a return on investment and has tangible benefits in terms of quality, productivity, time to market and customer satisfaction. For example, IBM Houston employed software inspections for the space shuttle missions: 85% of the defects were found by inspections and 15% were found by testing. There were no defects found on the space missions, and about 2 million lines of computer software were inspected. IBM, North Harbour in the UK quoted a 9% increase in productivity with 93% of defects found by software inspections.

Software inspections are useful for educating new employees on the product, and on the standards and procedures used in the organization. They ensure that knowledge is shared among the employees, rather than understood by just one individual. Inspections improve software productivity, as less time is spent in correcting defective software.

The cost of correction of a defect increases the later that it is identified in the lifecycle. Boehm [3] states that the *cost of correction of a requirements defect identified in the field is over 40 times more expensive than if it were detected at the requirements phase*, and so it is most economical to detect and fix the defect in phase. The cost of correction of a requirements defect identified at the customer site includes the cost of correcting the requirements, the cost of design, coding, unit testing, system testing and regression testing. It may be necessary to send an

engineer on site to fix the problem, and there may be hidden costs in the negative perception of the company with a subsequent loss of sales.

There is a powerful argument to identify defects as early as possible, and software inspections are a cost-effective way of doing this. There are various estimates of the *cost of poor quality* (COPQ) in an organization (Fig. 10.29), and some estimates suggest that it could be as high as 20–40% of sales. The exact calculation may be determined by a time sheet accountancy system, which details the cost of internal and external failure, and the cost of appraisal and prevention.

The return on investment from the introduction of software inspections may be calculated, and the evidence is that it leads to reductions in the cost of poor quality. Inspections provide a cost-effective way of improving quality and productivity.

6.3 Informal Reviews

This type of review involves reviewers sending comments directly to the author (e.g. email or written), and there is no actual review meeting. It is not as effective as the Fagan inspection process, but it helps in identifying some defects in the work products.

The author is responsible for making sure that the review happens and advises the participants that comments are due by a certain date. The author analyses the comments received, makes the required changes and circulates the document for approval. The activities are described in Table 6.1:

Comment:

The informal review process may help to improve quality in an organization. It is dependent on the participants adequately reviewing the deliverable and sending comments to the author. The author can only request the reviewer to send comments. There is no independent monitoring of the author to ensure that the review actually happens and is effective, and that comments are requested, received and implemented.

Table 6.1 Informal review

Step	Description
1.	The author circulates the deliverable (either physically or electronically) to the review audience
2.	The author advises the review audience of the due date for comments
3.	The due date for comments is typically one week or longer
4.	The author checks that all comments have been received by the due date
5.	The author contacts any reviewers who have not provided feedback and requests comments
6.	The author analyses all comments received and implements the appropriate changes
7.	The deliverable is circulated to the review audience for sign off
8.	The reviewers sign off (with any final comments) indicating that the document has been correctly amended by the author
9.	The author/project leader stores the comments received

6.4 Structured Walk-through

A structured walk-through is a peer review in which the author of a deliverable (e.g. a project document or actual code) brings one or more reviewers through the deliverable. The objective is to get feedback from the reviewers on the quality of the document or code and to familiarize the review audience with the author's work. The walk-through includes several roles, namely, the *review leader* (usually the author), the *author*, the *scribe* (may be the author) and the *review audience* (Table 6.2).

6.5 Semi-formal Review Meeting

A semi-formal review (a simplified version of the Fagan inspection) is a moderated review meeting chaired by the review leader. The author selects the reviewers and appoints a review leader (who may be the author). The review leader chairs the meeting and verifies that the follow-up activity has been completed. The author distributes the deliverable to be reviewed and provides a brief overview as appropriate. The material in this section is adapted from [4].

The review leader schedules the review meeting with the reviewers (with possible participation via a conference call). The review leader chairs the meeting and is responsible for keeping the meeting focused and running smoothly, resolving any conflicts, recording actions and completing the review form.

The review leader checks that all participants including conference call participants are present, and that all have done adequate preparation. Each reviewer is invited to give general comments, as this will determine whether the deliverable is

Table 6.2 Structured walk-throughs

Step	Description
1.	The author circulates the deliverable (either physically or electronically) to the review audience
2.	The author schedules a meeting with the reviewers
3.	The reviewers familiarize themselves with the deliverable
4.	The review leader (usually the author) chairs the meeting
5.	The author brings the review audience through the deliverable, explaining what each section is aiming to achieve, and requesting comments from them as to its correctness
6.	The scribe (usually the author) records errors, decisions and any action items
7.	A meeting outcome is agreed, and the author addresses all agreed items. If the meeting outcome is that a second review should be held then go to Step 1
8.	The deliverable is circulated to reviewers for sign off, and the reviewers sign off (with any final comments) indicating that the deliverable has been correctly amended by the author
9.	The author/project leader stores the comments and sign offs

ready to be reviewed, and whether the review should take place. Participants who are unable to attend are required to send their comments to the review leader prior to the review, and the review leader will present these comments at the meeting.

The material is typically reviewed page by page for a document review, and each reviewer is invited to comment on the current page. Code reviews may focus on coding standards, or on both coding standards and on finding defects in the software code. The issues noted during the review are recorded, and these may include items requiring further investigation.

The review outcome is decided at the end of the review (i.e. whether the deliverable needs a second review). The author then carries out the necessary corrections and investigation, and the review leader verifies that the follow-up activities have been completed. The document is then circulated to the review audience for sign off.

Comment:

The semi-formal review process works well for an organization when the review leader is not the author. This ensures that the review is conducted effectively, and that the follow-up activity takes place. It may work with the author acting as review leader provided the author has received the right training on software inspections and follows the review process.

The process for semi-formal reviews is summarized in Table 6.3. Figure 6.2 presents a template to record the issues identified during the review.

6.6 Fagan Inspections

The Fagan methodology (Table 6.4) is a well-known software inspection methodology. It is a seven-step process that includes planning, overview, preparation, an inspection meeting, process improvement, rework and follow-up activities. Its objectives are to identify and remove errors in the work products, and to identify any systemic defects in the processes used to create the work products.

The Fagan inspection process stipulates that requirement documents, design documents, source code and test plans all be formally inspected by experts independent of the author, and the inspection is conducted from different viewpoints such as requirements, design and test.

There are various roles defined in the inspection process, including the *moderator*, who chairs the inspection; the *reader*, who paraphrases the particular deliverable; the *author*, who is the creator of the deliverable; and the *tester*, who is concerned with the testing viewpoint. The inspection process will also consider whether the design is correct with respect to the requirements, and whether the source code is correct with respect to the design.

The goal is to identify as many defects as possible and to confirm the correctness of a particular deliverable. Inspection data is recorded and may be used to determine the effectiveness of the organization in detecting and preventing defects.

Table 6.3 Activities for semi-formal review meeting

Phase	Review task	Roles
Planning	Ensure document/code is ready to be reviewed Appoint *review leader* (may be author) Select reviewers with appropriate knowledge/experience and assign roles	Author Leader
Distribution	Distribute document/code and other material to reviewers (at least 3 days before the meeting) Schedule the meeting	Author Leader
Optional meeting	Give overview of deliverable to be reviewed Allow reviewers to ask any questions	Author Reviewers
Preparation	Read through document/code, marking up issues/questions Mark minor issues on their copy of the document/code	Reviewers
Review meeting	Review leaders chairs the meeting Explains purpose of the review and how it will proceed Set time limit for meeting Keep review meeting focused and moving Review document page by page Code reviews may focus on standards/defects Resolve any conflicts or defer as investigates Note comments/shortcomings on review form **Raise issues—(*Do not fix them*)** Present comments/suggestions/questions Pass review documents/code with marked up minor issues directly to the author Respond to any questions or issues raised Propose outcome of review meeting Complete review summary form/return to author Keep a record of the review form	Leader Reviewers
Post-review	Investigate and resolve any issues/shortcomings identified at review Verify that the author has made the required corrections	Author Leader

The moderator records the defects identified during the inspection, and the defects are classified according to their type and severity. The defect data may be entered into an inspection database to enable analysis to be performed and metrics to be generated. The severity of the defect is recorded, and the major defects are classified [e.g. according to the Fagan defect classification or some other scheme such as the *orthogonal defect classification* (ODC)].

The next section describes the Fagan inspection guidelines, which include recommendations on the time to spend on the various inspection activities. An organization may need to tailor the Fagan inspection process to suit its needs, and the tailored guidelines need evidence to confirm that they are effective.

6.6.1 Fagan Inspection Guidelines

The Fagan inspection guidelines are based on studies by Michael Fagan, and they provide recommendations on the time to spend on the various inspection activities. It

Date _____ **Deliverable** _____ **Version No.** _____ **#Reviews** _____
Author _____**Review Leader** _____
Reviewers_____
Page/Line No. **Description** **Action**

Unresolved Issued / Investigates
Issue **Reason unresolved** **Verified.**

Review Outcome (Tick)
No changes required □ Verification by Review Leader only □ Full review required □
Review incomplete □

Review Summary (Optional)
#Major Defects_____ # Minor Defects _____ Estimated Rework time _____
Hours Preparation _____ #Hours Review _____ Amount Reviewed _____

Fig. 6.2 Template for semi-formal review

Table 6.4 Overview Fagan inspection process

Activity	Role/Responsibility	Objective
Planning	Moderator	Identify inspectors and roles Verify material is ready for inspection Distribute inspection material Book a room for the inspection
Overview (Optional)	Author	Brief participants on material Give background information
Preparation	Inspectors	Prepare for the meeting and role Checklist may be employed Read through the deliverable and mark up issues/questions
Inspection meeting	Moderator/Inspectors	The moderator will cancel the inspection if inadequate preparation is done Time limit set for inspection Moderator keeps meeting focused The inspectors perform their roles Emphasis on finding defects not solutions Defects are recorded and classified Author responds to any questions The duration of the meeting is recorded An inspection outcome is agreed
Process improvement	Inspectors	Continuous improvement of development and inspection process The causes of major defects are recorded Root cause analysis to identify any systemic defect with development or inspection process Recommendations are made to the process improvement team
Rework	Author	The author corrects the defects and carries out any necessary investigations
Follow-up	Moderator/Author	The moderator verifies that the author has resolved the defects and investigations

is important that sufficient time is spent on the various inspection activities, and that the speed of the inspection is appropriate. We present the strict Fagan guidelines as defined by the Fagan methodology (Table 6.5), and more relaxed guidelines that have been shown to be effective in the telecommunications field (Table 6.6).

The effort involved in adherence to the strict Fagan guidelines is substantial, and this led to the development of tailored guidelines. The tailoring of any methodology requires care, and the effectiveness of the tailored process needs to be demonstrated by empirical evidence. (e.g. as a pilot prior to its deployment as well as quantitative data to show that the inspection is effective and results in a low number of escaped customer defects).

It is important to comply with the guidelines once they are deployed in the organization, and trained moderators and inspectors will ensure awareness and compliance. Audits may be employed to verify compliance.

The tailored guidelines are presented in Table 6.6.

Table 6.5 Strict Fagan inspection guidelines

Activity	Area	Amount/Hr	Max/Hr
Preparation time	Requirements	4 pages	6 pages
	Design	4 pages	6 pages
	Code	100 LOC	125 LOC
	Test plans	4 pages	6 pages
Inspection time	Requirements	4 pages	6 pages
	Design	4 pages	6 pages
	Code	100 LOC	125 LOC
	Test plans	4 pages	6 pages

Table 6.6 Tailored (Relaxed) Fagan inspection guidelines

Activity	Area	Amount/Hr	Max/Hr
Preparation time	Requirements	10–15 pages	30 pages
	Design	10–15 pages	30 pages
	Code	300 LOC	500 LOC
	Test plans	10–15 pages	30 pages
Inspection time	Requirements	10–15 pages	30 pages
	Design	10–15 pages	30 pages
	Code	300 LOC	500 LOC
	Test plans	10–15 pages	30 pages

6.6.2 Inspectors and Roles

There are four inspector roles identified in a Fagan Inspection and these include (Table 6.7):

6.6.3 Inspection Entry Criteria

There are explicit entry and exit criteria defined for the various types of inspections. These criteria need to be satisfied to ensure that the inspection is effective. The entry criteria (Table 6.8) for the various inspections are as follows:

6.6.4 Preparation

Preparation is a key part of the inspection process, as the inspection will be ineffective if the inspectors are insufficiently prepared. The moderator is required to cancel the inspection if any of the inspectors has been unable to do appropriate preparation.

Table 6.7 Inspector roles

Role	Responsibilities
Moderator	Manages the inspection process and ensures compliance to the process Plans the inspection and chairs the meeting Keeps the meeting focused and resolves any conflicts Keeps to the inspection guidelines Verifies that the deliverables are ready to be inspected Verifies that the inspectors have done adequate preparation Records the defects on the inspection sheet Verifies that the agreed follow-up work has been completed Skilled in the inspection process and appropriately trained Skilful, diplomatic and occasionally forceful
Reader	Paraphrases the deliverable and gives an independent view of it Actively participates in the inspection
Author	Creator of the work product being inspected Has an interest in finding all defects present in the deliverable Ensures that the work product is ready to be inspected Gives an overview to inspectors (if required) Participates actively during inspection and answers all questions Resolves all identified defects and carries out any required investigation
Tester	Role is focused on how the product would be tested Role often employed in requirements inspection/test plan inspection The tester participates actively in the inspection

Table 6.8 Fagan entry criteria

Inspection type	Entry criteria	Roles
Requirements	Inspector(s) with sufficient expertise available Preparation done by inspectors Correct requirements template used	Moderator/Inspectors
Design inspection	Requirements inspected and signed off Correct design template used to produce design Inspector(s) have sufficient domain knowledge Preparation done by inspectors	Moderator/Inspectors
Code inspection	Requirements/Design inspected and signed off Overview provided Preparation done by inspectors Code listing available Clean compile of source code Coding standards satisfied Inspector(s) have sufficient domain knowledge	Moderator/Inspectors
Test plan inspection	Requirements/Design inspected and signed off Preparation done by inspectors Inspector(s) have sufficient domain knowledge Correct test plan template employed	Moderator/Inspectors

6.6.5 The Inspection Meeting

The inspection meeting (Table 6.9) consists of a formal meeting between the author and at least one inspector. It is concerned with finding defects in the particular deliverable and verifying the correctness of the inspected material. The effectiveness of the inspection is influenced by

- The expertise and experience of the inspector(s)
- Preparation done by inspector(s)
- The speed of the inspection

These factors are quite clear since an inexperienced inspector will lack the appropriate domain knowledge to understand the material in depth. Second, an inspector who has inadequately prepared will be unable to make a substantial contribution during the inspection. Third, the inspection is ineffective if it tries to cover too much material in a short space of time. The moderator will complete the inspection form (Fig. 6.4) to record the results from the inspection.

The final part of the inspection is concerned with process improvement. The inspector(s) and author examine the major defects, identify the root causes of the defect and determine corrective action to address any systemic defects in the software process. The moderator is responsible for completing the inspection summary form and the defect log form, and for entering the inspection data into the inspection database. The moderator will give any process improvement suggestions directly to the process improvement team.

Table 6.9 Inspection meeting

Inspection type	Purpose	Procedure
Requirements	Find requirements defects Confirm requirements correct	Inspectors review each page of requirements and raise questions or concerns. Defects recorded by moderator
Design	Find defects in design Confirm correct (with respect to requirements)	Inspectors review each page of design (compare to requirements) and raise questions or concerns. Defects recorded by moderator
Code	Find defects in the code Confirm correct (with respect to design/reqs)	Inspectors review the code and compare to requirements/design and raise questions or concerns. Defects recorded by moderator
Test	Find defects in test cases/test plan Confirm test cases can verify design/requirements	Inspectors review each page of test plan/specification, compare to requirements/design and raise questions or concerns. Defects recorded by moderator

6.6.6 Inspection Exit Criteria

The exit criteria (Table 6.10) for the various inspections are as follows:

6.6.7 Issue Severity

The severity of an issue identified in the Fagan inspection may be classified as major, minor, a process improvement item or an item requiring further investigation. It is classified as *major* if its non-detection would lead to a defect report being raised later in the development cycle, whereas a defect report would generally not be raised for a *minor* issue. An issue classified as an investigate item requires further study, and an issue classified as process improvement is used to improve the software development process (Table 6.11).

Table 6.10 Fagan exit criteria

Inspection type	Exit criteria
Requirements	Requirements satisfy the customer's needs All requirements defects are corrected
Design	Design satisfies the requirements All identified defects are corrected Design satisfies the design standards
Code	Code satisfies the design and requirements Code satisfies coding standards and compiles cleanly All identified defects are corrected
Test	Test plan sufficient to test the requirements/design Test plan follows test standards All identified defects corrected

Table 6.11 Issue severity

Issue severity	Definition
Major (M)	A defect in the work product that would lead to a customer-reported problem if undetected
Minor (m)	A minor issue in the work product
Process improvement (PI)	A process improvement suggestion based on analysis of major defects
Investigate (INV)	An item to be investigated

6.6.8 Defect Type

There are several defect-type classification schemes employed in software inspections. These include the Fagan inspection defect classification (Table 6.12) and the orthogonal defect classification scheme (Table 6.13).

The orthogonal defect classification (ODC) scheme was developed at IBM [5], and a defect is classified according to three (orthogonal) viewpoints. The *defect trigger* is the catalyst that led the defect to manifest itself; the *defect type* indicates the change required for correction; and the *defect impact* indicates the impact of the defect at the phase in which it was identified. The ODC classification yields a rich pool of information about the defect, but effort is required to record this information. The defect-type classification is described in Table 6.13.

The defect impact provides a mechanism to relate the impact of the software defect to customer satisfaction. The impact of a defect identified pre-release is

Table 6.12 Classification of defects in Fagan inspections

Code inspection	Type	Design inspections	Type	Requirements inspections	Type
Logic (code)	LO	Usability	UY	Product objectives	PO
Design	DE	Requirements	RQ	Documentation	DS
Requirements	RQ	Logic	LO	Hardware interface	HI
Maintainable	MN	Systems inter-	IS	Competition	CO
Interface	IF	face		Analysis	
Data usage	DA	Portability	PY	Function	FU
Performance	PE	Reliability	RY	Software interface	SI
Standards	ST	Maintainability	MN	Performance	PE
Code	CC	Error handling	EH	Reliability	RL
Comments		Other	OT	Spelling	GS

Table 6.13 Classification of ODC defect types

Defect type	Code	Definition
Checking	CHK	Omission or incorrect validation of parameters or data in conditional statements
Assignment	ASN	Value incorrectly assigned or not assigned at all
Algorithm	ALG	Efficiency or correctness issue in algorithm
Timing	TIM	Timing/serialization error between modules, shared resources
Interface	INT	Interface error (error in communications between modules, operating system, etc.)
Function	FUN	Omission of significant functionality
Documentation	DOC	Error in user guides, installation guides or code comments
Build/Merge	BLD	Error in build process/library system or version control
Miscellaneous	MIS	None of the above

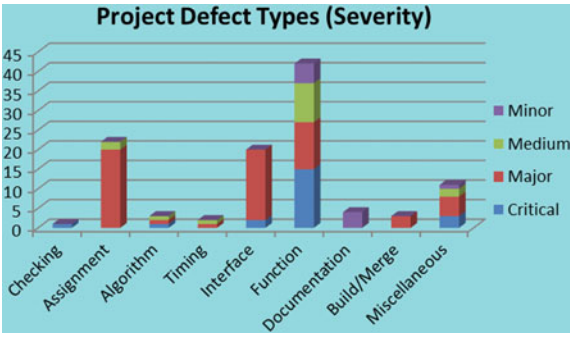

Fig. 6.3 Sample defect types in a project (ODC)

viewed as the impact of it being detected by an end-user, and for a customer-reported defect its impact is the actual information reported by the customer.

The inspection data is typically recorded in an inspection database, which allows analysis to be performed on the most common types of defects, and the preparation of action plans to minimize reoccurrence (Fig. 6.3). The frequency of defects per category is identified, and causal analysis is employed to identify preventive actions. Often, the most problematic areas are targeted first (as identified in a Pareto chart), and an investigation into the particular category is conducted. The action plans will identify actions to be carried out to improve the existing processes.

The ODC classification scheme may be used to give early warning on the quality and reliability of the software, as its use leads to an expected profile of defects for the various lifecycle phases. The actual profile may then be compared to the expected profile, and the presence of significant differences between these may indicate risks to quality.

For example, if the actual defect profile at the system test phase resembles the defect profile of the unit-testing phase, then it is likely that there are quality problems. This is clear since the unit-testing phase is expected to yield a certain pool of defects, with system testing receiving higher quality software with the defects found during unit testing corrected. Consequently, ODC may be applied to make a judgment of product quality and performance.

The inspection data will enable the *phase containment effectiveness* (PCE) metric to be determined (Fig. 10.19), and to determine if the software is ready for release to the customer.

6.7 Automated Software Inspections

Static code analysis is the analysis of software code without executing the code. It is usually performed with automated tools, and the sophistication of the tool determines the actual analysis done. Some tools may analyse individual statements or

Inspection Type _____ **Deliverable** _____ **Project** _____
Date _____ **Amount Inspected** _____ **Version No.** ____
Author_____ **Moderator**_____ **No. of Reviews** _____
Inspectors _____
#Hours Preparation _____ **# Hours Inspection** _____ **#Hours Rework** _____
Summary of Findings: **# Majors** _____ **# Minors** ____ **# PIs** _____ **# INVs** ____
ODC Summary (Majors): #CHK __ #ASS___ #ALG___ #TIM___ # INT__ #FUN___ # DOC___ # BLD___
_____
No. Page/Line No. Severity Type Description

Top 3 Root Causes of Major Defects / Process Improvement Actions
1.
2.
3.

Review Out come
No changes □ Verification by Moderator □ Full Review □ Review Incomplete □
Defects per KLOC _____ Defects per page _____ Verification of Rework _____
Date Verified _____ Inspection Data in Database ____

Fig. 6.4 Template for Fagan inspection

declarations, whereas others may analyse the whole source code. The objective of the analysis is to highlight potential coding errors early in the software development lifecycle.

These automated software inspection tools provide quality assessment reports on the extent to which the coding standards are satisfied. Many integrated development environments (IDEs) provide basic functionality for automated code reviews. These include Microsoft Visual Studio and Eclipse.

The LDRA Testbed tool automatically determines the complexity of the source code, and it provides metrics that give an indication of the maintainability of the code. A useful feature of the LDRA tool is that it gives a visual picture of system complexity, and it has a re-factoring tool to assist with reducing complexity. It automatically generates code assessment reports listing all of the files examined and provides metrics on the clarity, maintainability and testability of the code.

Compliance to coding standards is important in producing readable code and in preventing error-prone coding styles. There are several tools available to check conformance to coding standards including the LDRA TBvision tool, which has reporting capabilities to show code quality as well as fault detection and avoidance measures. It includes functionality to allow users to view the results presented intuitively in various graphs and reports. A selection of LDRA tools are presented in Chap. 17.

6.8 Review Questions

1. What are software inspections?
2. Explain the difference between informal reviews, structured walk-throughs, semi-formal reviews and formal inspections.
3. What are the benefits of software inspections?
4. Describe the seven steps in the Fagan inspection process.
5. What is the purpose of entry and exit criteria in software inspections?
6. What factors influence the effectiveness of a software inspection?
7. Describe the roles involved in a Fagan inspection.
8. Describe the benefits of automated inspections.

6.9 Summary

The objective of software inspections is to build quality into the software product, and there is clear evidence that the cost of correction of a defect increases the later in the software development cycle in which it is detected. Consequently, there is an economic argument to employing software inspections, as it is more cost effective to build quality in rather than adding it later in the development cycle.

There are several approaches to software inspections, and these vary in the level of formality employed. A simple approach consists of a walk-through of the document or code by an individual other than the author. The meeting is informal and usually takes place at the author's desk or in a meeting room, and the reviewer and author discuss the document or code informally.

There are formal software inspection methodologies such as the well-known *Fagan inspection* methodology. This approach includes pre-inspection activity, an inspection meeting and post-inspection activity. Several inspection roles are typically employed, including an *author* role, an *inspector* role, a *tester* role and a *moderator* role.

An organization will need to devise an inspection process that is suitable for its particular needs. The level of formality is influenced by its business, its culture and the potential impact of a software defect on its customers. It may not be possible to have all of the participants present in a room, and participation by conference call may be employed.

Software inspections play an important role in building quality into each phase, and in ensuring that the quality of the delivered product is good. The quality of the delivered software product is only as good as the quality at the end each phase, and therefore a phase should be exited only when the desired quality has been achieved.

The effectiveness of an inspection is influenced by the expertise of the inspectors, adequate preparation and speed of the inspection, and compliance to the inspection process. The inspection methodology provides guidelines on the inspection and preparation rates for an inspection, and guidelines on the entry and exit criteria for an inspection.

References

1. M. Fagan, Design and code inspections to reduce errors in software development. IBM Syst. J. **15**(3) (1976)
2. T. Gilb, D. Graham, *Software Inspections* (Addison Wesley, Boston, 1994)
3. B. Boehm, *Software Engineering Economics* (Prentice Hall, New Jersey, 1981)
4. F. O'Hara, *Peer Reviews—The Key to Cost Effective Quality*. (European SEPG, Amsterdam, 1998)
5. I. Bhandari, A case study of software process improvement during development. IEEE Trans. Softw. Eng. **19**(12), 1157–1170 (1993)