Static Testing

4

Key Topics

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

4.1 Introduction

Static testing (as distinct from *dynamic testing*) is a form of software testing that involves a systematic examination of the software code and documentation without execution of the code. It may be conducted manually or through the use of specialized software testing tools. There are several types of static testing such as code analysis, code reviews, structured walk-throughs, informal reviews and software inspections.

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 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.

G. O'Regan, *Concise Guide to Software Testing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-28494-7_4

There are several approaches to 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 (Fagan 1976) and the Gilb methodology (Gilb and Graham 1994). These methodologies include 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.

The Fagan inspection methodology was developed by Michael Fagan (Fig. 4.1) at IBM in the mid-1970s, and Gilb's approach to software inspections was developed by Tom Gilb 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. For example, an incorrect one-line change to telecommunication software could lead to a major telecommunication outage and significant disruption to customers.

Further, there may be financial impacts resulting from failure, as a service-level agreement provides details of the service level that will be provided, and the compensation for service disruption. Consequently, a telecommunication company needs to ensure that its software is fit for purpose, and a formal software inspection process is often employed to ensure that quality is built into the software. This means that requirement documents, design documents, software code, and test documents are all inspected, and these activities need to be included in the project schedule.

The 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 review process where the impact of a failure is less severe. It may not be possible to have all of the participants physically in a room, and it may be necessary to include some reviewers via conference call or a video link. It may be more appropriate to employ a structured walk-through or informal reviews for some organizations.

Fig. 4.1 Michael Fagan

Software inspections play an important role in building quality into the software, and the quality of the delivered software product is only as good as the quality at the end each phase, and so 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 by the inspectors, the speed in which the inspection is performed, and compliance with the inspection process. A formal inspection methodology provides guidelines on the inspection and preparation rates, and entry and exit criteria are defined for the inspection.

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

Next, we describe the benefits of software inspections, and we then discuss a simple review methodology, a structured walk-through, a semi-formal review process, and the Fagan inspection process.

4.2 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 (1981) states that the *cost of correction of a requirement defect identified in the field is over 40 times more expensive than if it were detected at the requirement phase*. Therefore, it is most economical to detect and fix defects in phase rather than correcting them later in the development cycle. The cost of correction of a requirement 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.

Therefore, it is important to identify defects as early as possible, and software inspections are a cost-effective way of doing this. The *cost of poor quality* (COPQ) in an organization (Fig. 1.9) may be determined, and it involves computing 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. That is, inspections provide a cost-effective way of improving quality and productivity.

4.3 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 helps in identifying some of the defects in the work products, but its success is dependent on the extent to which the reviewers are proactive in sending comprehensive comments to the author by the due date.

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 4.1.

COMMENT:
The informal review process 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 4.1 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, and the author may send a reminder to the reviewers
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 keeps a record of the comments received

4.4 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* (usually the author), and the *review audience*. It is described in more detail in Table 4.2.

4.5 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 of the material. This section is adapted from O'Hara (1998).

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 they have done sufficient preparation. Each reviewer is invited to give general comments, which will determine whether the deliverable is ready to be reviewed and whether the review should take place. Participants who

Table 4.2 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 author has correctly amended the deliverable
9.	The author keeps a record of the comments and sign-offs

are unable to attend are required to send their comments to the review leader prior to the review, and their comments are presented at the meeting.

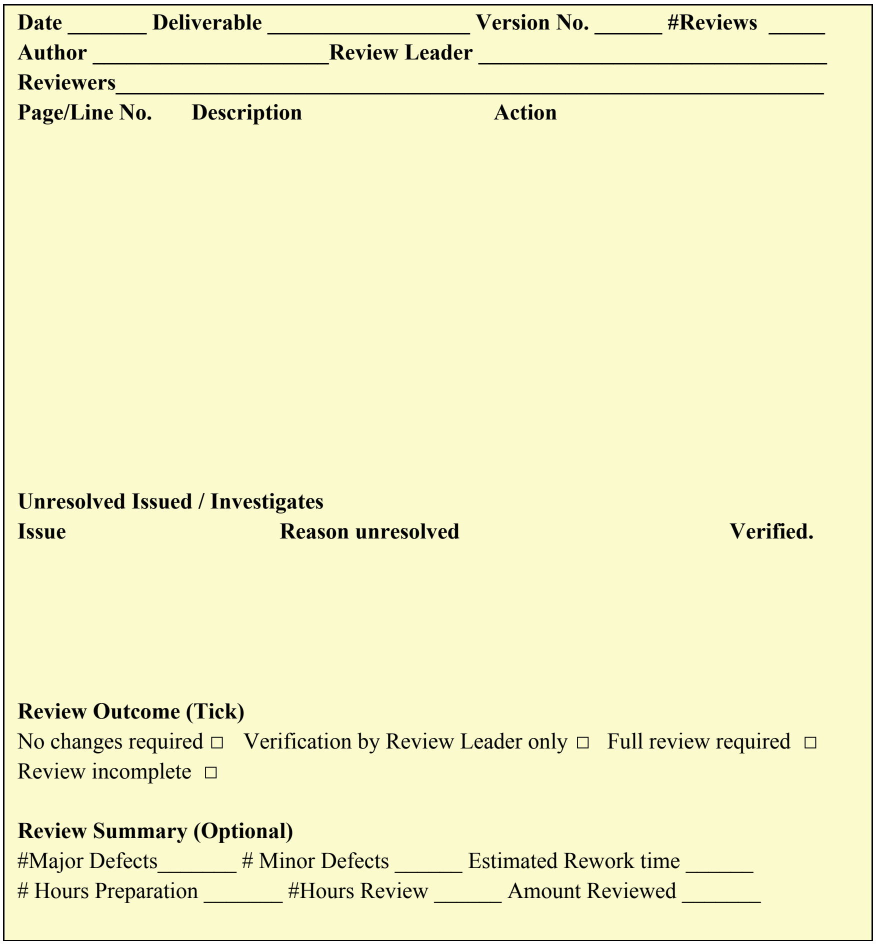
The material is typically reviewed page per page for a document review, and each reviewer is invited to comment on the current page. Code reviews may focus on coding standards or on both coding standards and finding defects in the software code. The issues noted during the review are recorded, and these 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 when the review leader is not the author, as otherwise there is a risk that the review may be ineffective and that the

Table 4.3 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 materials 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 chair the meeting Explain 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 coding standards and/or identifying defects Resolve any conflicts or defer to investigate Note comments/shortcomings on review form **Raise issues—(*do not fix them*)** Reviewers make comments/suggestions/questions Reviewers pass review documents/code with marked-up minor issues directly to the author Author responds to any questions or issues raised Propose outcome of review meeting Review leader completes review summary form Keep a record of the review form	Leader Reviewers Author
Post-review	Investigate and resolve any issues/shortcomings identified at review Verify that the author has made the required corrections	Author Leader



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. 4.2 Template for semi-formal review

follow-up activity may not be done. It may work with the author acting as review leader provided the author has received the right training on the review process and consistently follows the process.

The process for semi-formal reviews is summarized in Table 4.3. Figure 4.2 presents a template to record the issues identified during the review.

4.6 Fagan Inspections

The Fagan methodology (Fig. 4.3) is a well-known software inspection methodology that was developed at IBM in the mid-1970s. 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 are all formally inspected by experts independent of the author. The inspection is conducted from different viewpoints such as requirements, design, and test. (Table 4.4).

There are several 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 process will consider whether the design is correct with respect to the requirements and whether the source code is correct with respect to the design.

Fig. 4.3 Example of an inspection meeting (public domain)

Table 4.4 Overview Fagan inspection process

Activity	Role(s)	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 Inspectors	Brief participants on material Give background information to inspectors
Preparation	Inspectors	Prepare for the meeting and role Checklist may be employed Read through the deliverable and markup 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 is 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/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

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

The moderator records the defects identified during the inspection and classifies them 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. These guidelines are very strict, and an organization may need to tailor the Fagan inspection process to suit its needs. Any tailoring of the process and guidelines need empirical evidence to confirm that they are effective.

4.6.1 Fagan Inspection Guidelines

The Fagan inspection guidelines provide recommendations on the amount of time that should be devoted to the various inspection activities. It is important to spend sufficient time on preparation, and that the inspection meeting is not rushed and does not attempt to cover an excessive amount of material. We first present the strict Fagan guidelines as defined by the Fagan methodology (Table 4.5) and then consider more relaxed guidelines that have been shown to be effective in the telecommunication domain (Table 4.6).

The effort involved in adherence to the strict Fagan guidelines is substantial and led to the development of tailored guidelines. The tailoring of any methodology requires care, and empirical evidence is needed to demonstrate the effectiveness of

Table 4.5 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 4.6 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

the tailored process (e.g. a pilot prior to its deployment which includes quantitative data to show that the inspection is effective with a low number of escaped defects).

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

The tailored guidelines are presented in Table 4.6.

4.6.2 Inspectors and Roles

There are four inspector roles identified in a Fagan inspection (Table 4.7).

4.6.3 Inspection Entry Criteria

Entry criteria for the various types of inspections are specified in Table 4.8 and should be satisfied for an effective inspection.

Table 4.7 Inspector roles

Role	Responsibilities
Moderator	Manages the inspection process and ensures compliance with the process Trained in the inspection process and as a moderator Skilful, diplomatic, and occasionally forceful Plans the inspection and chairs the meeting Keeps to the inspection guidelines Verifies that the deliverables are ready to be inspected Verifies that the inspectors have done adequate preparation Keeps the meeting focused and resolves any conflicts Records the defects on the inspection sheet Verifies that the agreed follow-up work has been completed
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 required investigation
Tester	Role is focused on how the product would be tested Role often employed in requirement inspection/test plan inspection The tester participates actively in the inspection

Table 4.8 Fagan entry criteria

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

4.6.4 Preparation

Preparation is a key part of the process, as the inspection will be ineffective if insufficient time is devoted to understanding and reviewing the deliverables prior to the inspection meeting. Consequently, the moderator is required to cancel the inspection if the inspectors are insufficiently prepared.

4.6.5 The Inspection Meeting

The inspection meeting (Table 4.9) 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 its correctness. 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 is inadequately prepared will be unable to make a substantial contribution during the inspection. Third, the inspection is ineffective if it tries to cover

Table 4.9 Inspection meeting

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

too much material in a short space of time. The moderator will complete the inspection form to record the results from the inspection (Fig. 4.5).

The next 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 outcome of the inspection is agreed (e.g. inspect the material again or verification by the moderator). 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. The author then makes the agreed changes, and these are verified by the moderator (or a re-inspection).

4.6.6 Inspection Exit Criteria

The exit criteria (Table 4.10) for the various inspections are as follows.

4.6.7 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 (Table 4.11). 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

Table 4.10 Fagan exit criteria

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

Table 4.11 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

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.

4.6.8 Defect Type

There are several defect-type classification schemes employed in software inspections. These include the Fagan inspection defect classification (Table 4.12) and the orthogonal defect classification scheme (Table 4.13).

The orthogonal defect classification (ODC) scheme was developed at IBM (Bhandari 1993), and it classifies a defect into 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; the *defect impact* indicates the impact of the defect at the phase in which it was identified. The ODC 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 4.13.

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 4.12 Classification of defects in Fagan inspections

Code inspection	Type	Design inspections	Type	Requirement 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 Interface	IS	Completion analysis	CO
Data usage	DA	Portability	PY	Function	FU
Performance	PE	Reliability	RY	Software interface	SI
Standards	ST	Maintainability	MN	Performance	PE
Code Comment	CC	Error handling	EH	Reliability	RL
		Other	OT	Spelling	GS

Table 4.13 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 and 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

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 generally recorded in an inspection database, which allows analysis to be performed on the most common types of defects. The frequency of defects per category is identified, and causal analysis is employed to identify preventive actions (Fig. 4.4). The most problematic areas are targeted first (as identified in a Pareto chart), and an investigation into the causes of a particular category is conducted. Action plans will be prepared to improve the existing processes to prevent reoccurrence.

The ODC 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 significant differences 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 this may indicate quality problems. The

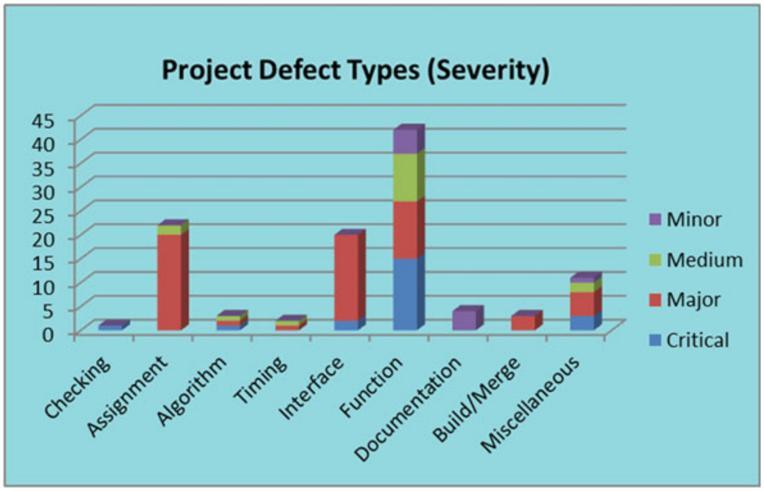


Fig. 4.4 Defect types in a project (ODC)

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 removed. Consequently, ODC may be applied to make a judgment of product quality and performance.

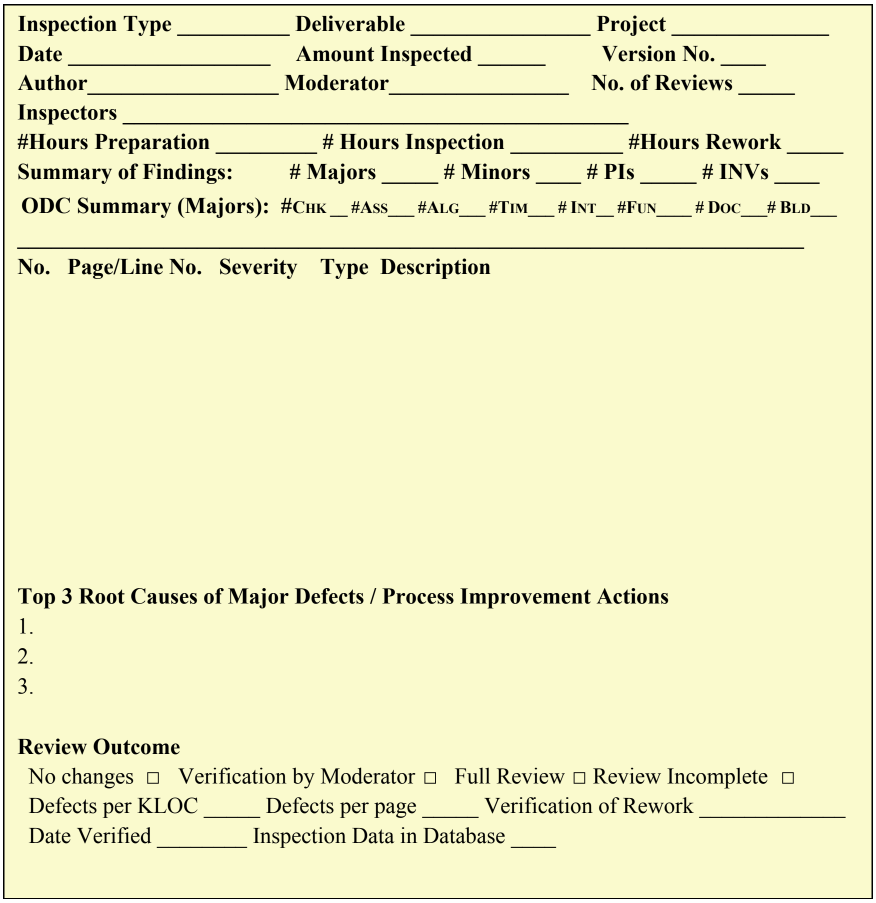
The inspection data will enable the *phase containment effectiveness* (PCE) metric to be determined (Fig. 9.10) and to determine if the software is ready for release to the customer.

4.7 Automated Code Inspections

Static code analysis is the analysis of software code without the execution of the code, and it is often performed with automated tools. The actual analysis done depends on the sophistication of the tool, with some tools analysing individual statements or 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 (Fig. 4.5).

Compilers find defects in the syntax of the software code, whereas static code analysis tools can analyse the software code to find more defects. Static code analysis provides early detection of defects prior to test execution, as well as making the software code easier to maintain. They also help in identifying defects that are hard to find in dynamic testing, and they may check for:

– Violation of coding standards
– Never-ending loops



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 Outcome
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. 4.5 Template for Fagan inspection

– Unreachable code
– Variables that are never used
– Referencing a variable with an undefined value
– Security vulnerabilities
– Parameter-type mismatch.

The automated software inspection tools provide quality assessment reports on the extent to which the coding standards are satisfied, as well as quality metrics on code complexity (Fig. 10.2). They provide warnings about potentially problems with the complexity of the design with metrics that have a high complexity measure. Many integrated development environments (IDEs) provide basic functionality for automated code reviews.

They provide metrics on the maintainability of the code, and some tools give a visual picture of system complexity and include a re-factoring feature to assist in reducing complexity. They automatically generate code assessment reports listing all of the files examined and provide metrics on the clarity, maintainability, and testability of the code.

The compliance with coding standards is important in producing readable and maintainable code, and in preventing error-prone coding styles. There are several tools available to check conformance to coding standards, which include reporting capabilities to show code quality as well as fault detection and avoidance measures.

4.8 Review Questions

1. What are software inspections?
2. Explain the difference between informal reviews, structured walk-throughs, semi-formal reviews, and formal inspections.
3. Explain the difference between static testing and dynamic testing
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.
9. What are the benefits of software inspections?

4.9 Summary

The objective of software inspections is to build quality into the software product, as the cost of correction of a defect increases later in the software development cycle in which it is detected. They make economic sense 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 reviews and inspections, including a walk-through of the document or code by an individual than the author. This 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.

The level of formality of an inspection process is influenced by the business and the potential impact of a software defect on its customers. It may not be possible to have all of the participants physically present in a room, and participation by conference call may be employed.

The effectiveness of an inspection is influenced by the expertise of the inspectors, adequate preparation, and speed of the inspection, and compliance with the inspection process.

Static code analysis is the analysis of software code without executing the code, and it is usually performed with specialized automated testing tools. The objective is to highlight potential coding errors early in the software development lifecycle.

References

Bhandari I (1993) A case study of software process improvement during development. IEEE Trans Softw Eng 19(12)

Boehm B (1981) Software engineering economics. Prentice Hall, New Jersey

Fagan M (1976) Design and code inspections to reduce errors in software development. IBM Syst J 15(3)

Gilb T, Graham D (1994) Software inspections. Addison Wesley, Boston

O'Hara F (1998) Peer reviews—the key to cost effective quality. European SEPG, Amsterdam