Software Inspections

6

Key Topics

Fagan Inspection
Gilb Inspections
Economic Benefits of Inspection
Inspection Guides
Entry and Exit Criteria
Informal Reviews
Prince 2 Quality Review
Agile Reviews

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 level of formality employed. A simple informal approach consists of a walkthrough 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 [20] and the Gilb methodology [24]. These methodologies include pre-inspection activity, an inspection meeting, and post-inspection

G. O'Regan, *Introduction to Software Quality*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-06106-1_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 the Gilb methodology was developed by Tom Gilb. The formality of the inspection methodology used by an organization is dependent on its type of business. The impact of a defect may have a major adverse effect on the customer's business: for example, an incorrect one-line change to telecoms software could create a major telecommunications outage and major disruption to customers. 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 inspection process tends to be employed. This means that requirement documents, high-level and detailed design documents, and code are inspected, and generally inspections are explicitly planned in the project schedule.

An organization will need to devise an inspection process which 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 adopt a formal approach such as the Fagan or Gilb methodology, or it may devise a less formal process tailored to its needs. It may not be possible to have all of the participants present in a room, and participation by conference call or video link may be employed. A formal inspection process may not suit some organization cultures, and an informal approach such as a structured walkthrough may be the adopted approach.

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, 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* role 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 slightly more formal inspection process is described, and finally the Fagan inspection process is described in detail.

6.2 Economic Benefits of Software Inspections

A software inspection program has tangible benefits in terms of productivity, quality, 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 two million lines of computer software were employed. 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 [7] 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 to achieve this.

There are various estimates of the *cost of poor quality* (COPQ) in an organization (Fig. 1.9), and estimates suggest that it may be 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 an introduction of software inspections may be calculated, and the evidence available suggests that they are a cost-effective way of improving quality and productivity.

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 1 week or longer.
4.	The author checks that all comments have been received by the due date
5.	Any reviewers who have not provided feedback are contacted by the author, and comments are requested.
6.	The author analyses all comments and implements the appropriate changes.
7.	The deliverable is circulated to the review audience for sign-off.
8.	The reviewers signoff (with any final comments) indicating that the document has been correctly amended by the author
9.	The author/project leader stores the comments received

6.3 Informal Reviews

This type of review involves reviewers sending comments directly to the author, 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 involved 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.*

6.4 Structured Walkthrough

A structured walkthrough 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 walkthrough 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).

Table 6.2 Structured walkthroughs

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 signoff and the reviewers signoff (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.

6.5 Semi-formal Review Meeting

A semi-formal review is a moderated review meeting chaired by the review leader. The leader may be the author, and the role involves chairing the meeting and verifying that the follow-up activity has been completed. The material in this section is adapted from [46].

The author selects the reviewers and appoints a review leader (who may be the author). The author distributes the deliverable to be reviewed, and provides a brief overview where appropriate.

The leader schedules the review meeting which includes 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 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 per page for a document review, and each reviewer is invited to comment on the current page. Code reviews may focus on coding standards only or may focus 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 this is verified by the review leader. The document is then circulated to the review audience for sign-off.

Table 6.3 Activities for semi-formal review meeting

Phase	Review task	Roles
Planning	Ensure document/code is ready to be reviewed	Author
	Appoint *review leader* (may be author)	Leader
	Select reviewers with appropriate knowledge/experience and assign roles	
Distribution	Distribute document/code and other material to reviewers at least 3 days before the meeting	Author
	Schedule the meeting	Leader
Optional meeting	Give overview of deliverable to be reviewed	Author
	Allow reviewers to ask any questions	Reviewers
Preparation	Read through document/code, marking up issues/questions	Reviewers
	Mark minor issues on their copy of the document/code	
Review meeting	Review Leaders chairs the meeting	Leader
	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)*	Reviewers
	Present comments/suggestions/questions	
	Pass review documents/code with marked up minor issues directly to the author	
	Respond to any questions or issues raised	Author
	Propose outcome of review meeting	Leader
	Complete review summary form/return to Author	
	Keep a record of the review form	
Post re-view	Investigate and resolve any issues or shortcomings identified at the review	Author
	Verify that the author has made the required corrections	Leader

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.

Deliverable _____ **Review Reference** ____
Date _____ **Version No.** ____
Author _____ **No. of Reviews** _____
Reviewers _____

Issue	**Page/Line No.**	**Description**	**Action No**

Unresolved Issued / Investigates

Issue	**Reason unresolved**	**Verified.**

Review Outcome (Tick one of the following)

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

6.6 Fagan Inspections

The Fagan methodology (Table 6.4) is a well-known software inspection methodology. It is a seven-step process and includes planning, overview, preparation, inspection meeting, process improvement, re-work, and follow-up activity. Its objectives are to identify and remove errors in the work products, and also 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, test, etc.

There are various roles defined in the inspection process, including the *moderator*, who chairs the inspection, the *reader*, who paraphrases the particular

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	Author	Brief participants on material.
		Give background information.
Preparation	Inspectors	Prepare for the meeting and role to be performed. 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
		A root cause analysis is performed to identify any systemic defect with the software development process or inspection process.
		Recommendations are made to the process improvement team.
Re-work	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.

deliverable, the *author*, who is the creator of the deliverable; and the *tester*, who is concerned with the testing viewpoint. The inspection process will consider whether a 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 are recorded and may be used to assess the effectiveness of the organization in detecting and preventing defects.

The moderator records the defects identified during the inspection, and the defects are classified according to their type and severity. Mature organizations typically enter defects into an inspection database to allow metrics to be generated, and to enable analysis to be performed. The severity of the defect is recorded, and the major defects are classified according to the Fagan defect classification scheme. Some organizations use other classification schemes, e.g., the *orthogonal defect classification* scheme (ODC).

The next section describes the Fagan inspection guidelines, and these include the recommended time to be spent on the various inspection activities. An organization may need to tailor the Fagan inspection process to suit its needs, and the recommended times in the Fagan process may need to be adjusted accordingly. The tailored guidelines will need empirical evidence to confirm that they are effective in defect detection.

6.6.1 Fagan Inspection Guidelines

The Fagan inspection guidelines are based on studies by Michael Fagan, and provide recommendations on the time to spend on the various inspection activities. The goal is to spend sufficient time to enable the inspection to be effective, and identify as many major defects as possible. Two tables are presented here: the strict Fagan guidelines as defined by the Fagan methodology (Table 6.5), and more relaxed guidelines that have been shown to be effective.

The effort involved in a strict adherence to the Fagan guidelines is substantial, and the tailored guidelines presented here have been employed in the telecoms domain. Empirical evidence of the effectiveness of the tailoring is not presented. Tailoring any methodology requires care, and the effectiveness of the tailoring should be demonstrated by a pilot prior to its deployment in the organization. This would generally involve quantitative data on the effectiveness of the inspection and the number of escaped customer reported 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 relaxed guidelines detailed in Table 6.6 do not conform to the strict Fagan inspection methodology.

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	125 LOC
	Test plans	4 pages	6 pages

Table 6.6 Tailored (Relaxed) Fagan inspection guidelines

Activity	Area	Amount/Hr	Max/Hr
Preparation	Requirements	10–15 pages	30 pages
Time	Design	10–15 pages	30 pages
	Code	300 LOC	500 LOC
	Test plans	10–15 pages	30 pages
Inspection	Requirements	10–15 pages	30 pages
Time	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 they are described in 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 for the various inspections are given in Table 6.8.

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

Table 6.9 Inspection meeting

Inspection type	Purpose	Procedure
Requirements	Find requirements defects. Confirm requirements correct and reflect customer's needs.	Inspectors review each page of requirements and raise questions or concerns. Defects recorded by Moderator
Design	Find defects in design and confirm its correctness 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 and confirm its correctness with respect to the design and requirements.	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 sufficient to verify the design/requirements.	Inspectors review each page of test plan/spec., compare to requirements/ design and raise questions or concerns. Defects recorded by moderator.

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 major 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. 6.3) 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.

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. 6.3 Template for Fagan inspection

Table 6.10 Fagan exit criteria

Inspection type	Exit criteria
Requirements	Requirements satisfy the customer's needs
	All requirements defects are corrected
Design inspection	Design satisfies the requirements.
	All identified defects are corrected
	Design satisfies the design standards
Code inspection	Code satisfies the design and requirements
	Code follows coding standards
	Code compiles cleanly
	All identified defects corrected
Test plan	Test plan sufficient to test the requirements
	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 (PI)	A process improvement suggestion based on analysis of
Improvement (PI)	major defects
Investigate (INV)	An item to be investigated. It is not clear whether it is a defect or not

6.6.6 Inspection Exit Criteria

The exit criteria for the various inspections are given in Table 6.10.

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 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).

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).

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 interface	IS	Competition analysis	CO
Interface	IF				
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 nor 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

The Orthogonal Defect Classification (ODC) scheme was developed at IBM [4], and a defect is classified according to three (orthogonal) viewpoints. Thee *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 requires effort 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 defect impact of a defect identified pre-release to the customer is viewed as the impact of the defect being detected by an end-user, and for a customer-reported defect, the impact is the actual information reported by the customer.

The inspection data is typically recorded in the inspection database; this will enable analysis to be performed on the most common types of defects, and enable actions to be identified to minimize reoccurrence. The data will enable the *phase containment effectiveness* (PCE) metric to be determined, and to determine if the software is ready for release to the customer.

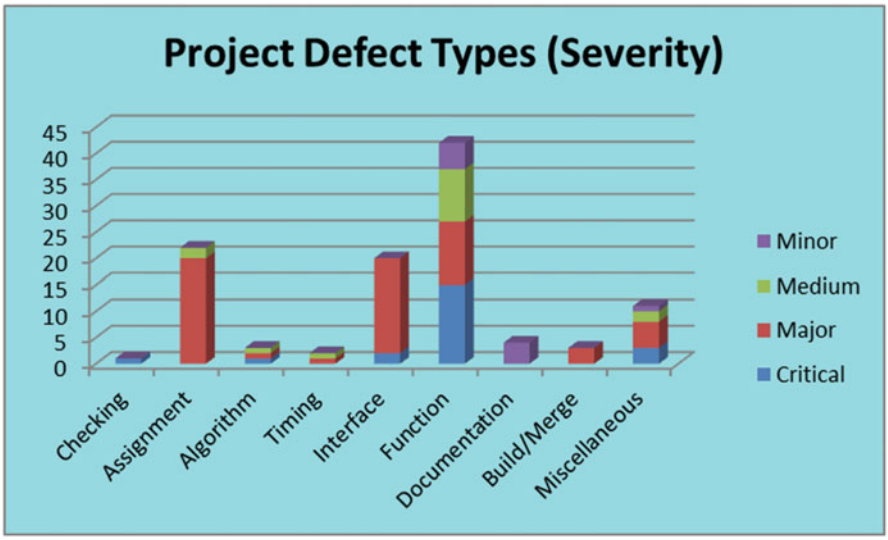

Fig. 6.4 Sample-defect types in a project (ODC)

The ODC classification scheme can give early warning on the quality and reliability of the software, as experience with the ODC classification scheme will enable an expected profile of defects to be predicted for the various phases. The expected profile may then be compared to the actual profile, and clearly it is reasonable to expect problems if the actual defect profile at the system test phase resembles the defect profile of the unit testing phase, as the unit testing phase is expected to identify a certain pool of defect types with system testing receiving higher-quality software with unit testing defects corrected. Consequently, ODC may be applied to make predictions of product quality and performance.

The project defects are classified according to some category scheme, for example, the defects may be categorized by the functional area in which they are identified, or via the ODC classification scheme as in Fig. 6.4. The frequency of defects per category is identified, and causal analysis employed to identify actions to prevent reoccurrence. Often the most problematic areas are targeted first (as in a pareto chart), and an investigation into the particular category is conducted. The action plans will identify and carry out improvements to existing processes.

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 actual analysis done depends on the sophistication of the tools. Some tools may analyze individual statements or

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

These tools provide automated software inspections, and provide quality assessment reports on the extent to which the 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.

6.8 Review Questions

1. What are software inspections?
2. Explain the difference between informal reviews, structured walkthroughs 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?
6. What factors affect 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 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 informal approach consists of a walkthrough 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 which 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 or video link may be employed. A formal inspection process may not suit some organization cultures, and an informal approach such as a structured walkthrough may be the adopted approach.

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.