The Prevention Principle:
Prevention is better than cure.
or
An ounce of prevention is worth a pound of cure.

The IEEE Definition of Inspection

The ANSI/IEEE Std. 729-1983 IEEE Standard Glossary of Software Engineering Terminology defines inspection as

> '...a formal evaluation technique in which software requirements, design, or code are examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems...'

It also defines the objective of software inspection as

> 'to detect and identify software elements defects. This is a rigorous, formal peer examination that does the following:
>
> (1) Verifies that the software element(s) satisfy its specifications.
> (2) Verifies that the software element(s) conform to applicable standards.
> (3) Identifies deviation from standards and specifications.
> (4) Collects software engineering data (for example, defect and effort data).
> (5) Does not examine alternatives or stylistic Issues.'
>
> (reference: ANSI/IEEE Std. 1028-1988, *IEEE Standard for Software Reviews and Audits*)

Felix Redmill (Redmill, 1988), Trevor Reeve (Reeve, 1991) and others report that work processes, other than those involved in software, have been applying 'idea Inspection' successfully over a wide range of documentation and drawings, at all stages of the product life cycle in commerce and industry.

1.2 Inspection and Other Review Techniques – a Comparison

1.2.1 Characteristics of Reviews and Walkthroughs

Traditionally, technical reviews and 'structured walkthroughs' have been held for software development products. They have been found to be 'helpful' in finding problems and improving the quality of software, but there is almost no substantial documentation on how

cost-effective these older methods are. However, these techniques are much less formal than Inspection, and are much less effective at identifying defects. They have gradually been replaced by formal Inspection, for example at IBM Rochester, Minnesota Labs (Lindner, 1992).

The Horses for Courses Principle:
Use walkthroughs for training, reviews for consensus, but use Inspections to improve the quality of the document and its process.

Reviews and walkthroughs are typically peer group discussion activities – without much focus on defect identification and correction. They are usually without the statistical quality improvement which is an essential part of Inspection. Walkthroughs are generally a training process, and focus on learning about a single document. Reviews focus more on consensus and buy-in to a particular document.

It may be wasteful to do walkthroughs or consensus reviews unless a document has successfully exited from Inspection. Otherwise you may be wasting people's time by giving them documents of unknown quality, which probably contain far too many opportunities for misunderstanding, learning the wrong thing and agreement about the wrong things.

Inspection is not an alternative to walkthroughs for training, or to reviews for consensus. In some cases it is a pre-requisite. The different processes have different purposes. You cannot expect to remove defects effectively with walkthroughs, reviews or distribution of documents for comment. However, in other cases it may be wasteful to Inspect documents which have not yet 'settled down' technically. Spending time searching for and removing defects in large chunks which are later discarded is not a good idea. In this case it may be better to aim for approximate consensus on technical issues first, and then Inspect the consensus documents. The educational walkthrough could occur either before or after Inspection (see Figure 1.5).

The description of 'Inspection' in other literature can be very misleading. For example, in the otherwise excellent Freedman and Weinberg (Freedman, 1982), no mention of metrics is made in the description of Inspection. They claim that Inspection differs from walkthroughs and other types of formal review by 'confining attention to a few selected aspects, one at a time.'

Other sources describe Inspection, or more accurately 'inspection', though in considerably less detail than in this book (Hollocker, 1990).

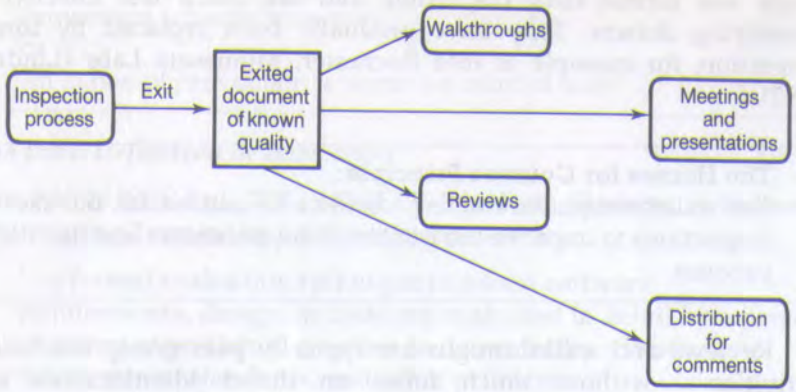

Figure 1.4 Inspection is not an 'alternative': Inspection first.

In other cases, some of the characteristics of Inspection are called by a different name. For example, Yourdon's 'Structured Walkthrough' emphasizes the purpose of detecting rather than correcting defects. It discourages managers from participation, and assigns similar roles and stages to the review process (Yourdon, 1989).

Definition, however, is not the only problem. The *IEEE Transactions on Software Engineering*, 1989 reports a recent industrial study in which it was found that 84% of organizations performed reviews or inspections, but 0% performed inspections entirely correctly.†

A detailed description of inspection is found in Humphrey (Humphrey, 1989) which describes a good inspection process for products, and which distinguishes inspections from management reviews, technical reviews and walkthroughs. Unfortunately, this book mistakenly believes that inspections are more relevant to detailed design and code, and so misses the most effective use of the technique. Otherwise, the rest of the description of the inspection process is well worth reading.

Humphrey places the use of inspection at Level 3 of the SEI process maturity framework, as a technique used by organizations with a defined (as well as repeatable) process. Garcia (Garcia, 1991) also makes the point that Inspections can be set up at the start of a measurement program.

Expected benefits of Inspection, when compared with reviews, walkthroughs or other types of inspection (with a small 'i') are:

† D.B. Bisant and J.R. Lyle, A Two-Person Inspection Method to Improve Programming Productivity, *IEEE Transactions on Software Engineering*, **15** (10), Oct 89, 1294–1304.

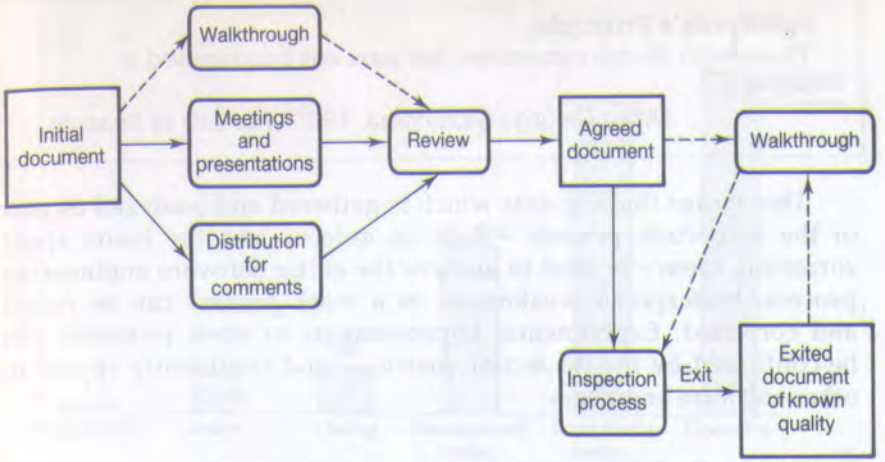

Figure 1.5 Inspection is not an 'alternative': Inspection last.

- measurably higher product *quality;*
- greater *productivity* from people in development and maintenance;
- shorter and more predictable *development times*.

A paradoxical problem with Inspection is that even a pale imitation of the 'real thing', such as a walkthrough, can be quite effective at improving software quality by finding some important defects. Even doing Inspections badly can give quite good results. This is particularly evident when no type of review at all has been performed to date. However, the 'good' really does work against the 'best' in this case, as the benefits from doing real Inspection properly would be far greater. The tragedy is that many people are satisfied with the 10% improvement, and never explore the potential for 100% or more improvement.

Trevor Reeve in Chapter 14 (Case Studies) of this book discusses more of the differences between the various 'cousins' of review techniques.

1.2.2 Statistical Quality Improvement

The fundamental difference between Inspection and other review methods is that Inspection provides a tool to help improve the entire development process, through a well-known quality engineering method, 'statistical process control' (Godfrey, 1986).

Santayana's Principle:
Those who do not remember the past are condemned to relive it.
(After George Santayana, 1905, *The Life of Reason*)

This means that the data which is gathered and analyzed as part of the Inspection process – data on defects, and the hours spent correcting them – is used to analyze the entire software engineering process. Widespread weaknesses in a work process can be found and corrected. Experimental improvements to work processes can be confirmed by the Inspection metrics – and confidently spread to other software engineers.

1.2.3 Early Defect Detection and Correction

A second important difference between Inspection and other forms of review is the emphasis on the formal written logging of potential defects in the software task document (requirements, design, test plans, test cases, source code, for example) at the earliest possible moment. This is followed by equally formal emphasis on the correction of defects before the documents in question can pollute further development or production and test efforts.

Experience has shown that the cost of executing software tests to catch and correct problems is at least an order of magnitude greater than if such items are found and corrected earlier by using Inspection (see Figure 1.6).

The Sewing Principle:
A stitch in time saves nine.

1.3 Inspection Compared with Testing

1.3.1 What Inspection and Testing have in Common

Inspection and testing both aim at evaluating and improving the quality of the software engineering product before it reaches the customers. The purpose of both is to find and then fix errors, defects and other potential problems.

Inspection and testing can both be applied early in software development, although Inspection can be applied earlier than test.

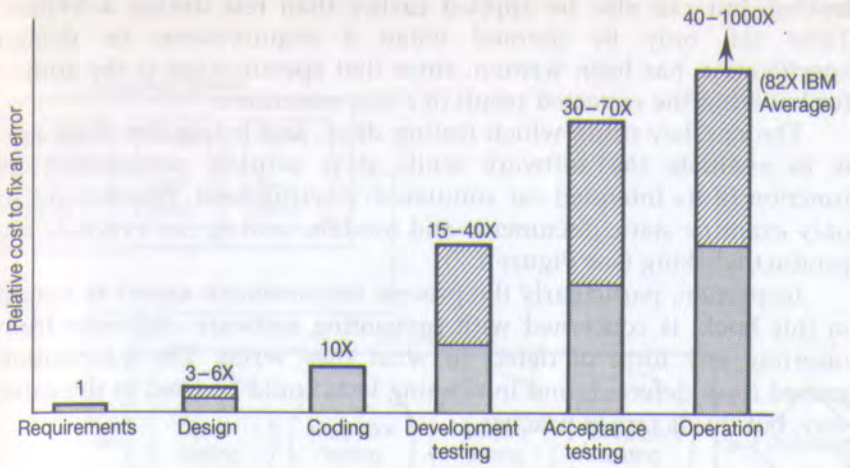

Figure 1.6 The cost of fixing errors escalates as we move the project towards field use. From an analysis of sixty-three projects cited in Boehm: *Software Engineering Economics* (Boehm, 1981).

Both Inspection and test, applied early, can identify defects which can then be fixed when it is still much cheaper to do so.

Inspection and testing can be done well or badly. If they are done poorly, they will not be effective at finding defects, and this causes problems at later stages, test execution, and operational use.

We need to learn from both Inspection and test experiences. Inspection and testing should both ideally (but all too rarely in practice) produce product-defect metrics and process-improvement metrics, which can be used to evaluate the software development process. Data should be kept on defects found in Inspection, defects found in testing, and defects which escaped both Inspection and test, and were only discovered in the field. This data would reflect frequency, document location, severity, cost of finding, and cost of fixing.

There is a trade-off between fixing and preventing. The metrics should be used to fine-tune the balance between the investment in the defect detection and defect prevention techniques used. The cost of Inspection, test design, and test running should be compared with the cost of fixing the defects at the time they were found, in order to arrive at the most cost-effective software development process.

1.3.2 Differences between Inspection and Testing

Inspection can be used long before executable code is available to run tests. Inspection can be applied much earlier than dynamic

testing, but can also be applied earlier than test design activities. Tests can only be defined when a requirements or design specification has been written, since that specification is the source for knowing the expected result of a test execution.

The one key thing which testing does, and Inspection does not, is to evaluate the software while it is actually performing its function in its intended (or simulated) environment. Inspection can only examine static documents and models; testing can evaluate the product working (see Figure 1.7).

Inspection, particularly the process improvement aspect as taught in this book, is concerned with preventing software engineers from inserting any form of defect in what they write. The information gained from defects found in running tests could be used in the same way, but this is rare in practice.

1.3.3 The Relationship between Inspection and Testing

Testing and Inspection are not mutually exclusive alternatives. Both have a role, and the best of both worlds is achieved using each where it is most appropriate. Test documentation is just as prone to defects as any other document, so it is essential that test documents are Inspected.

Defining tests early means Inspecting test plans early. Professional software testing practices include:

- the planning and design of testing from the beginning of the software development life cycle;
- acceptance tests defined by users at the same time as user requirements are defined;
- system tests defined during requirements analysis – integration tests defined during architectural design;
- unit tests defined during detailed design;
- execution of unit tests;
- execution of integration tests;
- execution of system tests;
- execution of acceptance tests;
- evaluation of test effectiveness.

In fact, it is even more effective to define tests *before developing* the life cycle deliverable, as argued by Bill Hetzel (Hetzel, 1984).

The consequence of early test planning is that Inspection can be effectively used to improve the quality of the test planning documents and the test planning process at early stages of a project. This will help avoid 'last minute before deadline' problems in getting projects successfully out of the door.

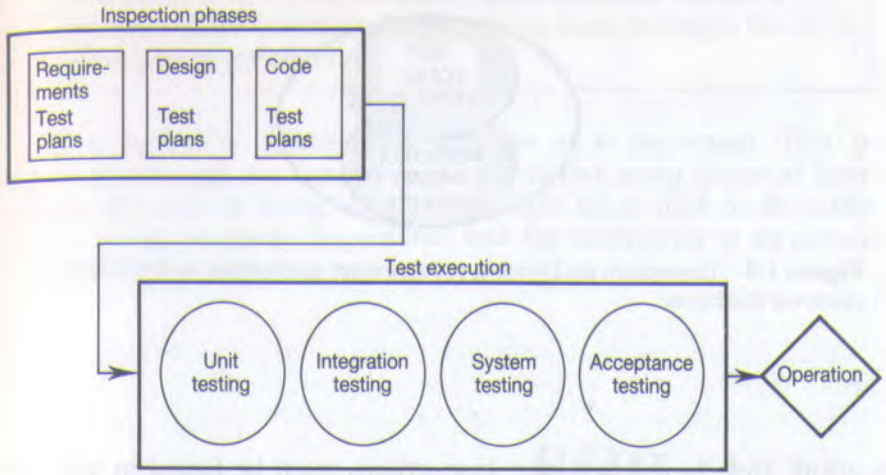

Figure 1.7 Inspection can only examine static documents. Testing can look at the end product.

Inspecting tests can save a lot of time (85% at IBM for example) in the test running phases of the development (Larson, 1975). Test documents are also used in Inspection. For example, requirements-based test cases or plans would be a source document for the Inspection of a design implementing those requirements. Does the design correspond to the planned tests?

> **Experience data:**
> Unisys Inspection Experience
> One organization (Unisys) which implemented better testing practices found that the number of defects was reduced by 50% by introducing five levels of testing. Their next step was to implement Inspection, which gave them an eventual result of a four-fold improvement in four years (Ulmer, 1991).

Inspection does not replace testing. They both perform some unique functions, which neither can replace for the other (see Figure 1.8). Inspection finds defects not found in testing, and testing finds problems not found in Inspection (Sutton, 1992). What testing *could* find, but Inspection finds *first*, probably gives a saving in correction

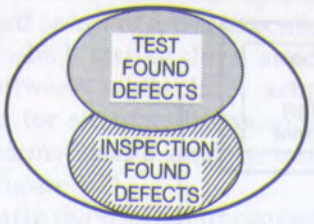

Figure 1.8 Inspection and testing complement each other in the defect removal business.

effort. Defects inserted *after* Inspections, must be found in test. The defect insertion rate during Inspection corrections was reported at IBM by Fagan (Fagan, 1986) as one out of six correction attempts.

AT&T (1986, *op cit.*) correctly classified Inspection as a form of testing. The mechanics are the same, but Inspection does not use a machine to work out what the result of an input (source) and a rule (program) would be in terms of expected 'output' (or 'task product'). This human effort is more costly than if a computer did it, but for many kinds of Inspection no computer is yet capable of making the necessary judgments.

It should be noted that the first of *either* of these two techniques which is actually applied always seems more effective, simply because there are, at the earlier stage, more defects available to find.

1.3.4 Inspection and Defect-finding Tools

Static analysis tools can find some types of defects, such as data-flow or control-flow errors and anomalies. Control-flow problems include unreachable code or infinite loops. Data-flow errors include variables declared and not used, or variables with a value stored twice without being referenced in between.

Software tools are very effective at finding certain types of defects. People are good at making trivial errors such as misspellings or punctuation errors, but are very poor at finding them. Software tools are much faster and more accurate in identifying these, but they cannot find all types of defect. Do not Inspect for things which can more easily be found at the stage you are at by automated tools. Inspect for those things which the tools can't detect at that stage.

> **The Tools for Fools Principle:**
> Automated tools *should* be used to find problems, but not if you must delay detection until the fixing costs outweigh the advantage of automation.

The notion of which stage you are at is important. This is because although we can automatically detect some errors at later stages, the cost of fixing the defects might be so high as to easily justify using people to Inspect and find the same error at an earlier stage of the development process.