8.1.2 Code Walk-Throughs

A code walk-through is an informal technique for analysis of the code. A code walk-through of a module is undertaken after the coding of the module is complete. In this technique, after a module has been coded, members of the development team select some test cases and simulate execution of the code by hand. Even though a code walk-through is an informal analysis technique, several guidelines have been evolved over the years for making this naive but useful analysis technique more effective. Of course, these guidelines are based on personal experience, common sense, and several subjective factors, and hence should be considered more as examples than rules to be applied dogmatically. Some of these guidelines are given below:

- The team performing a code walk-through should not be either too big or too small. Ideally, it should consist of between three to seven members. [3 7] ✓
- Discussions should be focussed on the *discovery* of errors and not on how to fix the discovered errors.
- In order to foster cooperation and avoid the feeling among the engineers that they are being evaluated, managers should not participate in the discussions.

8.1.3 Code Inspections

In contrast to code walk-throughs, code inspections aim explicitly at the discovery of commonly made errors. In other words, during code inspection the code is examined for the presence of certain kinds of errors, in contrast to the hand simulation of code execution as done in code walk-throughs. For instance, consider the classical error of writing a procedure that modifies a formal parameter while the calling routine calls that procedure with a constant actual parameter. It is more likely that such an error will be discovered by looking for it in the code, rather than by simply hand simulating the execution of the procedure. Most software development companies collect statistics to identify the type of errors most frequently committed. Such a list of commonly committed errors can be used during code inspections to keep a look-out for possible errors.

The following is a list of some classical programming errors which can be looked for during code inspection:

- Use of uninitialized variables.
- Jumps into loops.
- Nonterminating loops. (Infinite loops)
- Incompatible assignments.
- Array indices out of bounds.
- Improper storage allocation and deallocation.
- Mismatches between actual and formal parameters in procedure calls.
- Use of incorrect logical operators or incorrect precedence among operators.
- Improper modification of loop variables.
- Comparison of equality of floating point values, etc.

Adherence to coding standards is also checked during code inspections.

8.1.4 Software Documentation

We have already seen that when we develop a software product, we not only develop the executable files and the source code but also develop various kinds of documents such as users' manual, software requirements specification (SRS) document, design document, test document, installation manual, etc. All these documents are a vital part of any good software development practice. Good documents enhance understandability and maintainability of a software product.

Different types of software documentation can be broadly classified into:

- internal documentation, and
- external documentation (supporting documents).

Internal documentation is the code comprehension features provided as part of the source code itself. Internal documentation is provided through appropriate module headers and comments embedded in the source code. Internal documentation is also provided through the use of meaningful variable names, code indentation, code structuring, use of enumerated types and constant identifiers, use of user-defined data types, etc. Most software development organizations usually ensure good internal documentation by appropriately formulating their coding standards and coding guidelines.

External documentation is provided through various types of supporting documents such as users' manual, software requirements specification document, design document, test documents, etc. A systematic software development style ensures that all these documents are produced in an orderly fashion.

An important feature of good documentation is consistency. If the different documents are not consistent, a lot of confusion is created for somebody trying to understand the product. Also, all the documents for a product should be up-to-date. Even if only a few documents are not up-to-date, they create inconsistency and lead to confusion.