

1 Walkthroughs and inspections

An approach that doesn't make use of a computer at all in trying to eradicate faults in a program is called inspection or a walkthrough.

1.1 The individual and the error

Programmers are often seen as loners. Given a clear specification, a programmer often carries out the complete process of program design, coding and testing entirely on their own. Programmers are seen as low-profile techies in contrast to the articulate extrovert systems analysts. Thus a program is sometimes seen as a personal work of art; the creation of an individual programmer. These attitudes denied that “two heads are better than one,” that through discussion with others we can produce better work.

The common experience that someone else can spot errors better than the author led to the invention of the *structured walkthrough*. Credit for its invention belongs to Weinberg, in his book *The Psychology of Computer Programming*. Weinberg suggested that programmers see their programs as an extension of themselves. He suggested that we get very involved with our own creations and tend to regard them as manifestations of our own thoughts. We are unable to see errors in our own programs, since to do so would be to find a fault in ourselves, and this, apparently, is unacceptable to us. The term for this is *cognitive dissonance*. The solution is to seek help with fault finding. In doing this we relinquish our private relationship with our work. Programming becomes *ego-less programming*. This is a completely informal technique, carried out by colleagues in a friendly manner. It is not a formalized method carried out at fixed times and made into a rigid procedure of the organization. Indeed, to formalize ego-less programming would be to destroy its ethos and therefore its effectiveness. In a walkthrough or inspection, someone simply studies the program listing (along with the specification) in order to try to see bugs. It works better if the person doing the inspecting is not the person who wrote the program. This is because people tend to be blind to their own errors. So if you get a friend or a colleague to inspect your program, it is extraordinary to witness how quickly someone else sees an error that has been defeating you for hours. Studies also show that different people tend to uncover different errors. This further suggests the use of team techniques.

1.2 Structured walkthroughs

A structured walkthrough is simply the term for an organized meeting at which a program (or some other product) is examined by a group of colleagues. The major aim of the meeting is to try to find bugs which might otherwise go undetected for some time. (There are other goals, which are explained later.) The word “structured” simply means “well organized.” The term “walkthrough” means the activity of the programmer explaining step-by-step the working of his/her program. The reasoning behind structured walkthroughs is just this that by letting other people look at your program, errors will be found much more quickly.

To walkthrough a program you need only:

the specification

the text of the program on paper.

In carrying out a walkthrough, a good approach is to study it one procedure at a time. Some of the checks are fairly straightforward:

variables initialized

loops correctly initialized and terminated

procedure calls have the correct parameters.

Another check depends on the logic of the procedure. Pretend to execute the procedure as if you were a computer, avoiding following any calls into other procedures. Check that the logic of the method achieves its desired purpose.

During inspection you can also check that:

variable and procedure names are meaningful

indentation is clear and consistent.

The prime goal of a walkthrough is to find bugs, but checking for a weallness in style may point to a bug.

The evidence from controlled experiments suggests that walkthroughs are a very effective way of finding errors. In fact walkthroughs are at least as good a way of identifying bugs as actually nrrning the program (doing testing).

Although structured walkthroughs were initially used to find bugs in program code, the technique is valuable for reviewing the products at every stage of development - the tequlrements specification, a formal software specification, architectural design, program design, the code, the test data, the results of testing, the documentation.

There are several key points in organizing walkthroughs successfully:

Gauge the size and membership of the group carefully so that there are plenty of ideas, but so that everyone is fully involved.

Expect participants to study the material *prior* to the meeting.

Concentrate attention on the *product* rather than the person, to avoid criticizing the

author.

Limit the length of the meeting, so that everyone knows that it is business-like.

Control the meeting with the aid of agreed rules and an assertive chair.

Restrict the activity to *identifling* problems, not solving them.

Briefly document the faults (not the cures) for later reference.

The benefits of structured walkthroughs can be:

1. Software quality is improved because
 - more bugs are eliminated
 - the software is easier to maintain, because it is clearer.
2. Programmer effort is reduced because
 - specifications are clarified before implementation
 - errors are detected early and so costly rework is avoided
 - the time spent at the meeting (and in preparation for it) is more than repaid in time saved.
3. Meeting deadlines is improved because
 - visibility of the project is better (so potential catastrophes are prevented)
 - major errors are avoided early
4. Programmer expertise is enhanced because
 - everyone learns from everyone else.
5. Programmer morale is Improved because
 - people gain satisfaction from better work
 - people get to find out what is going on
 - people enjoy the discussions with colleagues.

Of course walkthroughs do mean that the individual has to be relaxed about presenting their work to colleagues.

1.3 Inspections

Inspections are similar to structured walkthroughs - a group of people meet to review a piece of work. But they are different from walkthroughs in several respects. Checklists are used to ensure that no relevant considerations are ignored. Errors that are discovered are classified according to type and carefully recorded on forms. Statistics on errors are computed, for example in terms of errors per 1000 lines of code. Thus inspections are not just well organized, they are completely formal. In addition management is informed of the results of inspections, though usually they do not attend the meeting. Thus inspections are potentially more threatening to the programmer than walkthroughs.

There are other; minor; differences between inspections and walkthroughs. Normally there are only four members in an inspection team:

- the moderator; who coordinates activities

- the designer who designed the program component being inspected

- the programmer

- the tester a person who acts as someone who will be responsible for testing the component.

The essence of inspections is that the study of products is carried out under close management supervision. Thus inspections are overtly a mechanism for increased control over programmers' work, in a similar fashion to the way that quality control is carried out on a factory floor. Many programmers would feel threatened in this situation and become defensive, perhaps trying to hide their mistakes. Perhaps this worsens the discovery of errors, and makes programming a less enjoyable activity

From an organizational point of view keeping records of faults discovered during inspections provides information to predict the quality of the software being written. Also, by highlighting common mistakes it can be used to improve programmers' self awareness and thereby improve their skills.