# Chapter 9: Testing the system

1. The major function of the first pass is building a symbol table, while the object of the second pass is to produce machine code. These are the two major functions that should be tested separately (although these functions could be broken down further into subfunctions, which could in turn be tested separately). The assembler should be clearly divided into components that implement these two functions. The part of the assembler that builds the symbol table could be tested by running it on an assembly language and capturing the output, which is a symbol table. The symbol table could then be checked for correctness. Once the first part of the assembler is well tested (on several different assembly programs), the second part could be tested by running it on symbol tables which are known to be correct, and then checking the resulting machine code for correctness. Finally, the entire system would be tested by running it on an assembly language program and then checking the machine code that results. The build plan could be designed similarly, with the symbol table building functions coded, integrated, and tested first, then the code generation subsystem, then finally the entire system.
2. This type of test would be considered a function test because it is performed in order to ensure that all the functions of the compiler work properly. The requirements being tested are functional requirements, not non-functional requirements, so it is not a performance test. It is not an acceptance test because it is not performed by the customer against customer expectations (other than in the sense that the customer probably expects it to be certified). It is not an installation test because it is not concerned with differences between the development and use environments.
3. One example is the availability of developers. If many of the developers assigned to implement the system are only available for a restricted calendar period, then the number of builds may be restricted so that the developers can code intensively, uninterrupted, for the time that they are available.
4. Causes:

   1. The first four characters of the command are "LINE."
   2. The command contains exactly two parameters separated by a comma and enclosed in parentheses.
   3. B = 0
   4. B <> 0

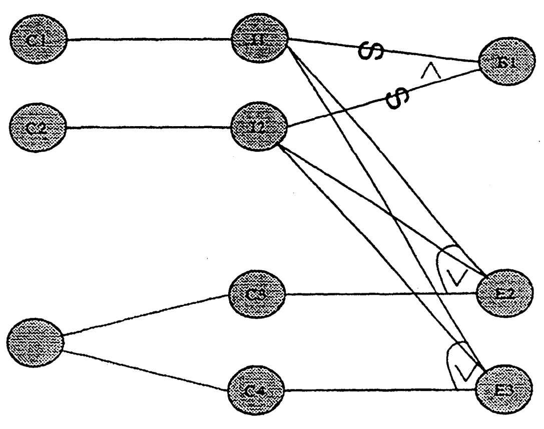
   Effects:

   1. System outputs "Improperly formed command."
   2. System outputs "D and E undefined."
   3. System outputs D and E, which are real numbers

   Intermediate effects:

   1. The command is syntactically valid.
   2. The operands are syntactically valid.

   Cause-and-effect graph:



5. Performance testing is concerned with testing non-functional requirements that specify things like speed, accuracy, and security. It is easy to specify these types of attributes ambiguously because the desire is always for the highest speed, the most accuracy, and the highest security possible. However, in order to be testable, very specific acceptance levels must be specified. For example, the requirement that the system must have "an acceptable turnaround time" is not testable because it does not give precise limits for the turnaround time. A testable requirement would be that the system "performs function X on up to 500K bytes of data in no more than 5 seconds.

6. Performance tests for a word processing system might include usability tests, i.e. tests that ensure compliance with non-functional requirements concerned with user-friendliness. Also, there might be requirements concerning the acceptable turnaround time for users, which would be tested using timing tests. Timing might also be a subject of performance tests for a payroll system, as well as security, to ensure that no one without the proper authorization can access payroll records. Security also would be a concern in performance testing an automated bank teller system, as well as recovery, timing and stress tests to ensure that the system will not get bogged down during peak times. A water quality monitoring system might be tested for the accuracy of its calculations, as well as environmental tests for speed and reliability under adverse conditions, such as natural disasters. It might also be tested for compatibility with the hardware devices (e.g. sensors) that it interacts with. Environmental tests would also be applied to a power plant control system, as well as timing and accuracy tests.

7. In the simplest case, such a system could have two configurations, one for the single-user version and one version for multiple users. There could be more than two configurations if there are multiple platforms or hardware that the software must operate with. In the simple case, the set of tests could be organized into two groups: one set that tests core functionality that is applicable to both versions, and one set that tests only the functions having to do with multiple users, e.g. timing, etc.

8. Some issues to consider:

   - what hardware devices are present on the airplane that were not available or were simulated in the development environment?
   - what regression tests should be performed?
   - what does the user consider to be the most important aspects to test?
   - what extreme conditions (e.g. weather) could be present at the installation site that were not duplicated in the development site?

9. Regression testing after the software was modified would have been the best way to detect this fault. An installation test at the Guam airport would probably also have uncovered the error.

10. If the "device" in question is a human being (e.g. for medical equipment), then it is not feasible to use an actual human for testing the software and some sort of simulator must be used. Also, if the device is inaccessible to the development environment (e.g. aboard a spacecraft in orbit) and is too large and expensive to duplicate for software testing, then a simulator must be used. Another example is when the software needs to be tested when the device is responding to some catastrophic situation with severe consequences (e.g. a device detecting a nuclear power plant meltdown). A system simulator might also be needed in this last situation so that the meltdown could be simulated (and not actually happen), as well as the system's response (shutting down the plant).

11. The form is very complete. It includes when and how the problem manifested itself, all of the events surrounding the discrepancy that might have an effect on why it occurred and how it might be fixed, as well as administrative information that will help in keeping the discrepancy process organized. Other administrative information that might be helpful would be the name of the system or subsystem being tested, the version and the release. Information about how the fault was found and fixed (e.g. components that needed to be modified, effort to isolate and fix the fault, etc.) would be included in a fault report form.

12. **Stress tests** would not be particularly useful, as this type of system would not be subject to severe unpredictable variations in activity. **Volume tests** might be applicable if there is likely to be very large amounts of payroll data. **Configuration tests** would not be applicable unless the system has several configurations. **Compatibility tests** would be required with any other systems, e.g. for inputting the payroll data and depositing the pay in employees' bank accounts. **Regression tests** are required if the system is replacing an existing system. **Security tests** are definitely needed to ensure the availability, integrity and confidentiality of payroll data. **Timing tests** need to be performed if there is a concern about acceptable response time for user requests. **Environmental tests** are probably not necessary for

this type of system. **Quality tests** might be appropriate to evaluate the system's reliability, maintainability and availability. **Recovery tests** would be appropriate to see if the system recovers properly from failures. **Maintenance tests** might be required if diagnostic tools and procedures are provided along with the software. **Documentation tests** would be needed to ensure that we have written the required documents. **Human factors tests** would also be important to evaluate display screens, messages, report formats, etc.

13. Installation test would be important in this situation because the installation sites may be different in significant ways from the development site. It would be important to check that each configuration has been calibrated correctly for the size of each factory, as well as to evaluate any other environmental differences.

14.
```
    Step 1:    Determine values of A and B which indicate a low water
               level.
    Step 2:    Type "LEVEL(A,B)" where A and B are the values
               calculated in Step 1.
               The system displays "LEVEL = SAFE."
    Step 3:    Determine values of A and B which indicate a safe water
               level.
    Step 4:    Type "LEVEL(A,B)" where A and B are the values
               calculated in Step 3.
               The system displays "LEVEL = SAFE."
    Step 5:    Determine values of A and B which indicate a high water
               level.
    Step 6:    Type "LEVEL(A,B)" where A and B are the values
               calculated in Step 5.
               The system displays "LEVEL = HIGH."
    Step 7:    Type "LEVL(A,B)" where A and B are any values.
               The system displays "INVALID SYNTAX."
    Step 8:    Type "LEVEL(A,$)" where A is any value.
               The system displays "INVALID SYNTAX."
```

15. The two sets of definitions are fundamentally different. Shooman's measures give a sense of the average behavior of a system, while the probability measures more completely describe system behavior. The probabilities can be converted into "mean time" measures by choosing the time at which the probability is 0.5, but the "mean time" measures cannot be converted into probabilities in general.

16. All of these people share some responsibility for the failure. In addition, those involved in writing the requirements (on which the test plan should have been based) share some responsibility, as well as the designers and coders who all could have considered that case.

17. Several factors must be considered in making this decision:

    - what would be the consequences of a failure in the system?
    - what would be the consequences of not deploying the system?
    - what is the cost of building the system in the first place?
    - are there other ways to address the problem, or parts of the problem?

18. The managers are, in part, responsible because they needed to review the procedures used by the V&V team to ensure that they were thorough. The V&V team itself is largely responsible because they missed something important that later caused a failure. The designers, coders, and testers might also share the responsibility depending on the ultimate source of the error.

19. The sum of $F(t)$ and $R(t)$ is always 1, so if reliability is increasing, the values of $R(t)$ will increase and the values of $F(t)$ will decrease for all values of $t$. In other words, the graphs of the two functions will become farther apart.

20. Some CM issues:

    - What parts of the system are common to the two different versions?
    - Which tests are equally applicable to both versions?
    - Of the tests that are applicable to both, which are the most important ones to be used for regression testing?

- What dependencies exist between common components and components that are specific to each of the two versions?

A good configuration management strategy might have helped Wind River to more easily port their operating system because it would have made clear which parts of the system would need to be modified and how the new modified system should be tested.

21. In analyzing testing principles and methods, it is important to be able to tell when a test case or testing activity reveals a fault in the software. In order to do this, it is necessary to distinguish the behavior exhibited in the test environment from the desired behavior. In complex systems, erroneous behavior may appear similar to the desired behavior, so an oracle is required to distinguish the two.

22. A possible plan is outlined below. It starts with the basic pricing functions based on the time and day of the advertisement. It adds functionality by considering, in turn, other factors that affect price and strategy:

| *Spin* | *Functions* | *Test start* | *Test end* |
|---|---|---|---|
| 0 | Basic Pricing (based on day and time) | 1 January | 31 January |
| 1 | Ratings | 28 February | 15 March |
| 2 | Restrictions | 25 April | 5 June |
| 3 | Opposition schedules | 10 June | 20 June |