Assignment 1

Paul Collado

Department of Computer Science

Montclair State University

CSIT-415

Software engineering II

03/04/2015

1. Discus the specific kinds of test which might have exposed the problems that caused each of the failures of Ariane 5?

Failure 1: Since all three of the variables were left unprotected, this was definitely an algorithm error. This could have been spotted by a Statement testing

Failure 2: Since the code was left to run for longer than needed then this could was a computation and precision error and could have been spotted by an All-uses testing

Failure 3: This could error was due to the reuse of an old specification, this was a specification error. This could have been avoided by a revision on the specification

Failure 4: This error was due to a bad behavior error catching from the Ariane programme. This could have been avoided by an All-computational-uses/some-predicate-uses testing

Failure 5: A piece of software was not re-tested, because it did not have a trajectory data in its specification, Again a Specification error.

Failure 6: This failure was due to the bad design decision of not including the SRI in the loop. This is a design error and could have been avoided by definition-use-path testing.

1. Chapter 7, question 9: What are the advantages and disadvantages of using the same standardized language or tools across all applications in your organization?

Some advantages may be:

* Provides an advantage for testing the application
* Expected input is responsible for possible output
* The type, name and purpose of every variable is well defined
* Expected modification or extensions can be easy
* If code the code is reusable for client side and server side then I will improve efficiency

Some disadvantages may be:

* The client side coding sometimes may be suitable for script or markup language but server side might be using business logic and might be using other language than that of client side language

1. Chapter 7, question 11: How can control flow be documented for an object-oriented program?

An object oriented system enhances the reusability of the software applications and its properly prepared documentation increases the user readability. Preparing documentation for object oriented program requires the description of reused components in different manner than that of the description of maintenance part of the application. The static and dynamic state of the system which includes control flow should also be distinguished. The documentation scheme for an object oriented program is given as below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Overview** | **external view** | **internal view** |
| **Static view** | Static Overview | Class interface description | Class implementation description |
| **Dynamic view** | Dynamic overview | Task interface description | Task implementation description |

So, the documentation4 of object oriented program will have six categories as described in the above given picture name as static overview, class interface description, class implementation description, dynamic overview, task interface description and task implementation description

1. Chapter 8, question 4: Suppose a program contains N decision points, each of which has two branches. How many test cases are needed to perform path testing on such a program? If there are M choices at each decision point, how many test cases are needed for path testing? Can the program’s structure reduce this number? Give an example to support your answer

As the program is having two N number of decision points and each decision point is having two branches, thus, the total test plan required will be **2N + 1**

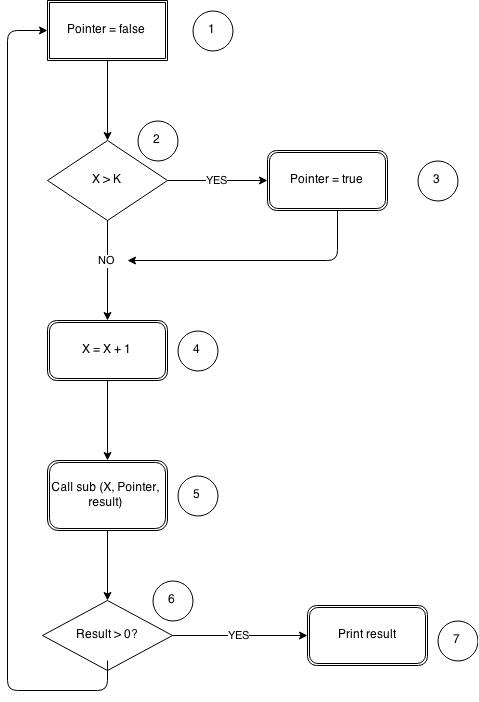
As every decision point is having M number of choices, then, the total test cases required for testing will be 2MN. The total test cases are dependent to the structure of the program. The example to show this is below:



On the diagram the rent is shown. On the bases of testing approach and structure of the program total number of test cases can be reduced

1. Chapter 8, question 5: Consider a program flow diagram as a directed graph in which the diamonds and boxes of the program are nodes, and the logic flow arrows between them are directed edges. For example, the program in figure 8.7 can be graph as shown in figure 8.21. Prove that statement testing of a program is equivalent to finding a path in the graph that contains all nodes of the graph. Prove that branch testing is equivalent to finding the set of paths whose union covers the edges. Finally, prove that path testing is equivalent to finding all possible paths through the graph.

A flow diagram is used to show the flow logic from on code segment to another or from one function to other. The flow diagram with directed edges is a given below:



1. Chapter 8, question 7: figure 8.22 illustrates the component hierarchy in a software system. Describe the sequence of test for integrating the components using a bottom-up approach, a top-down approach, a modified top-down approach, a big-band approach, a sandwich approach, and a modified sandwich approach

Testing approaches

Maintainability demonstration could be conducted as the major part of the evaluation effort and system testing. It thus also provides the assessment for the most of the factors that supports logistics and relates and impacts the parameters of maintainability and the downtime of the items.

The system major functional elements are validated along with the system associated rates of failures. Thus the various sequential test plans are considered that are validated and is thus tested and which reaches on a particular decision making concept.

The maintainability demonstration is accomplished usually in the later phase of the detailed design and the development phase and must be conducted within the environment that would simulate the maintenance and the operational environment that is planned for the item. It could be varied easily when it is dependent on the requirements of system and the objectives of test.

The testing approach in which all the sub-models of the application are combined and testing is performed, is known as integration testing. The approaches are:

The testing approach, the integrated sub-models of the application are tested first and the related modules are tested step by step is known as top down approach

The testing approach, in which the sub-modules are tested first and then combined for testing is known as bottom up approach

The combination of bottom up and top down testing is known as sandwich testing

The testing approach, in which most of the modules of the application are combined and testing is done is known as big bang approach