Antibugging strategy in Software Testing

Homework #2

Submitted by : Karumanchi Sravanthi
skaruman@cstp.umkc.edu

Question #1

Develop a set of guidelines for antibugging.

Answer #1

Good design dictates that error conditions must be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error occurs. Yourdon calls this approach ***antibugging***.

Guidelines for antibugging:

1. *Antibugging Strategy must be used right from the start*

   These strategies should be incorporated early into the design phase rather than wait till the project is almost done. It becomes difficult to start testing when the project is completely done as it might need some redesigning and it would be too late to startover. The best way to build an efficient and reliable product is to prevent errors in the initial stages of the development. Software quality is the prime priority from the initial stages.

   Error conditions must be anticipated and must be taken care of. Errors must be tested so that the error handling paths can be successfully terminated from processing when they occur or could also be rerouted.

   The antibugging techniques are basically verification and validation techniques for user data, passed parameters, initial values to the parameter, data from the disk or files, user authentication and authorization data.

2. *Error descriptions have to be intelligible and should indicate the possible cause of error*

   Error description should try to explain the cause of the error and try and convey the location of the error as it helps to try and remove the error. For example when it says " Error: never reach this part of the program", it does not help as we do not know where the error has occurred and it becomes ambiguous.

3. *It should be able to identify error-prone areas*

   Testing is an area where one finds errors in the program. We must be able to differentiate the data that is invalid and the one that is valid. Doing this wrong leads to faulty test cases and thus improper incorporation of error handling paths.

4. *Errors should be categorized and handled uniformly throughout the system*

   The errors can be categorized by their source or type of handling they require. There should be uniformity in handling the error. For example if there is a way an error is indicated during input, then it must be maintained in a similar fashion throughout.

5. *Tackle unanticipated errors*

   Generally we anticipate errors based on the test data and then take action. But sometimes there is a possibility that we might have to face unexpected or unanticipated errors. This could be due to lack of exhaustive testing or because of some subtle conditions that are met. Consequently when an unexpected error occurs, the existing antibugging framework cannot detect the source or its corresponding action. Hence we should establish some steps to take care of these unanticipated errors.

6. *Inadequacies in development of test cases*

- *Inadequate protection against corrupted data*- There is no guarantee that data on the disk is reliable. Someone can edit it or change the data.
- *Inadequate test for user input*- It is not fair that users are told to enter the data as input in a specific format.
- *Inadequate test of passed parameters*- A subroutine must make sure that the data passed to it are valid and should not assume that it is always called correctly.
- *Inadequate protection against using it in a wrong way*- Users may purposely use it with bad input or try to trigger errors.
- *Inadequate use of version*- If there are multiple versions of the executable code then there is a possibility that different versions are used especially when the executable code is placed in multiple files.
- *Inadequate initial state validation*- If a program should start with all zeroes then it should not assume that it is zeroes but perform a check.

7. *Error condition processing should be carefully coded*

   If it is possible for the program to correct the error, then the error handling code should be written with lot of care. Calling wrong error handling routine may cause more problems than the actual errors. So the error condition processing must be properly coded.

8. *Adequate documentation of error handling routines and the error messages should be done.*

   It is very important to document all error messages produced by the system. Complete documentation of the error-handling routines is a must with the functional documentation of the system.

Question #2

Discuss advantages of using this technique

Answer #2

The advantages of this technique are:

1. Testing of the program is easier since the error handling paths are provided in the program itself. The program defends itself from bad input, wrong stimuli from other parts of the program and other exceptions occurring.
2. The behavior of the program becomes more predictable even in cases of bad inputs. This many times prevents unexpectedly crashing or termination of the program. This increases the user-friendliness of the program.
3. Description of errors is much more comprehensible. The error messages can become a good indication toward how to rectify the error and what is the location of the error.
4. Removes discrepancies between errors reported and actual error occurred.
5. Implicit assumptions are tested explicitly.
6. Prevents error conditions from intervening the system prior to appropriate error handling.

Question #3

Discuss disadvantages of using this technique.

Answer #3

The disadvantages of using this technique are:

1. Code, which is not related to functional requirements of the program, has to be incorporated in the program. Redundant code is incorporated to check system state after modification.

2. Test cases have to be designed to check all the error-handling paths. If one doesn't, then the path may fail when it is invoked, exacerbating an already tough situation.
3. Time for program/system development increases. The overall time period of development increases.
4. Currently no strategies or guidelines are defined for anti-bugging. Thus it is completely a designers intellect and skill that helps in effective anti-bugging.
5. Various tests have been performed to check if it is implemented correctly.

References:

1. Software Engineering – A practitioner's approach *Roger S. Pressman* McGraw-Hill International Edition. Fifth Edition
2. A Review of Automated Debugging Systems: Knowledge, Strategies, Techniques *Mireille Ducasse, Anna-Maria Emde* ACM Proceedings http://dev.acm.org/pubs/articles/proceedings/soft/55823/p162-ducasse/p162-ducasse.pdf
3. www.QAcity.com Organized resources
4. IEEE Software Best Practices January 1996
5. SCISM Home Page
6. Object Oriented Analysis by Peter Coad and Edward Yourdon
7. Parasoft Press Releases – Dr. Adam Kolawa