Manual testing

Compare with Test automation.

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user whereby they use most of the application's features to ensure correct behavior. To guarantee completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases.

Contents

Overview

A key step in the process is testing the software for correct behavior prior to release to end users.

For small scale engineering efforts (including prototypes), exploratory testing may be sufficient. With this informal approach, the tester does not follow any rigorous testing procedure, but rather explores the user interface of the application using as many of its features are the possible, using information gained in prior tests to intuitively derive additional tests. The success of exploratory manual testing relies heavily on the domain expertise of the tester, because a lack of knowledge will lead to incompleteness in testing. One of the key advantages of an informal approach is to gain an intuitive insight to how it feels to use the application.

Large scale engineering projects that rely on manual software testing follow a more rigorous methodology in order to maximize the number of defects that can be found. A systematic approach focuses on predetermined test cases and generally involves the following steps.[1]

1. Choose a high level test plan where a general methodology is chosen, and resources such as people, computers, and software licenses are identified and acquired.
2. Write detailed test cases, identifying clear and concise steps to be taken by the tester, with expected outcomes.
3. Assign the test cases to testers, who manually follow the steps and record the results.
4. Author a test report, detailing the findings of the testers. The report is used by managers to determine whether the software can be released, and if not, it is used by engineers to identify and correct the problems.

A rigorous test case based approach is often traditional for large software engineering projects that follow a Waterfall model.[2] However, at least one recent study did not show a dramatic difference in defect detection efficiency between exploratory testing and test case based testing.[3]

Testing can be through black-, white- or grey-box testing. In white-box testing the tester is concerned with the execution of the statements through the source code. In black-box testing the software is run to check for the defects and is less concerned with how the processing of the input is done. Black-box testers do not have access to the source code. Grey-box testing is concerned with running the software while having an understanding of the source code and algorithms.

Static and dynamic testing approach may also be used. Dynamic testing involves running the software. Static testing includes verifying requirements, syntax of code and any other activities that do not include actually running the code of the program.

Testing can be further divided into functional and non-functional testing. In functional testing the tester would check the calculations, any link on the page, or any other field which on given input, output may be expected. Non-functional testing includes testing performance, compatibility and fitness of the system under test, its security and usability among other things.

Stages

There are several stages. They are:

Unit Testing
: This initial stage in testing normally carried out by the developer who wrote the code and sometimes by a peer using the white box testing technique.

Integration Testing
: This stage is carried out in two modes, as a complete package or as an increment to the earlier package. Most of the time black box testing technique is used. However, sometimes a combination of Black and White box testing is also used in this stage.

System Testing
: In this stage the software is tested from all possible dimensions for all intended purposes and platforms. In this stage Black box testing technique is normally used.

User Acceptance Testing
: This testing stage carried out in order to get customer sign-off of finished product. A 'pass' in this stage also ensures that the customer has accepted the software and is ready for their use.

Release or Deployment Testing
: Onsite team will go to customer site to install the system in customer configured environment and will check for the following points:

1. Whether SetUp.exe is running or not.
2. There are easy screens during installation
3. How much space is occupied by system on HDD
4. Is the system completely uninstalled when opted to uninstall from the system.

Advantages of Manual Testing

- Low-cost operation as no software tools are used
- Most of the bugs are caught by manual testing
- Humans observe and judge better than the automated tools

Comparison to automated testing

Test automation may be able to reduce or eliminate the cost of actual testing. A computer can follow a rote sequence of steps more quickly than a person, and it can run the tests overnight to present the results in the morning. However, the labor that is saved in actual testing must be spent instead authoring the test program. Depending on the type of application to be tested, and the automation tools that are chosen, this may require more labor than a manual approach. In addition, some testing tools present a very large amount of data, potentially creating a time consuming task of interpreting the results.

Things such as device drivers and software libraries must be tested using test programs. In addition, testing of large numbers of users (performance testing and load testing) is typically simulated in software rather than performed in practice.

Conversely, graphical user interfaces whose layout changes frequently are very difficult to test automatically. There are test frameworks that can be used for regression testing of user interfaces. They rely on recording of sequences of keystrokes and mouse gestures, then playing them back and observing that the user interface responds in the same way every time. Unfortunately, these recordings may not work properly when a button is moved or relabeled in a subsequent release. An automatic regression test may also be fooled if the program output varies significantly.

See also

- Test method
- Usability testing
- GUI testing
- Software testing
- Codeless test automation

References

1. ANSI/IEEE 829-1983 IEEE Standard for Software Test Documentation
2. Craig, Rick David; Stefan P. Jaskiel (2002). *Systematic Software Testing*. Artech House. p. 7. ISBN 1-58053-508-9.
3. Itkonen, Juha; Mika V. Mäntylä; Casper Lassenius (2007). "Defect Detection Efficiency: Test Case Based vs. Exploratory Testing" (https://web.archive.org/web/20161013081515/https://pdfs.semanticscholar.org/9ed8/52540fc2c77dab4152b14b4381b9dc124c69.pdf) (PDF). *First International Symposium on Empirical Software Engineering and Measurement*: 61–70. doi:10.1109/ESEM.2007.56 (https://doi.org/10.1109%2FESEM.2007.56). ISBN 978-0-7695-2886-1. Archived from the original (http://www.soberit.hut.fi/jitkonen/Publications/Itkonen_Mäntylä_Lassenius_2007_ESEM.pdf) (PDF) on October 13, 2016. Retrieved January 17, 2009.