OPEN ACCESS Freely available online

Advances in Medical Ethics

Short Communication

Approaches to Negative Testing Using Mutation Testing

Jaweed Abdul*

Department of Pharmaceutical Chemistry, Jahangirnagar University, Savar Union 1342, Bangladesh

DESCRIPTION

Negative testing addresses the crucial issue of determining a system's capacity to deal with unforeseen circumstances. If left unattended, such circumstances may result in system failures that, in certain cases, may have catastrophic effects. In order to generate test cases that support negative testing, this paper provides a mutation testing-based methodology. Application of this strategy can deliver test cases that successfully evaluate a wide range of unexpected circumstances in a methodical and human-unbiased manner. As a result, it can help a tested system improve. The article offers a general architecture for the production and execution of the test cases, explicitly specifies mutation operators used to manage the generation process, and discusses how to analyze results.

The development of reliable software systems that satisfy their users depends in large part on testing. Positive and negative testing should both be done in a complete test. The many actions involved in determining if a system satisfies all stakeholders needs make up positive testing. There are several testing strategies that may be used to do this objective. Even if the methods differ in specifics, they all generally revolve around the idea of picking test cases based on some kind of specification, running the test cases against the system, and comparing the results to what was expected and defined. Results of carefully conducted positive testing are typically regarded as a clear and reliable indication of how accurate a system is in terms of complying with the required specifications [1-3].

A system that functions properly under normal circumstances could, nevertheless, nonetheless not be able to adequately manage some unusual, unexpected, and generally undesirable scenarios. The system might crash or fail to produce the desired results if certain conditions are not handled properly. This could have catastrophic repercussions, such as harm to property or loss of life. Therefore, it is necessary to do a negative test as well. The goal of negative testing is to evaluate a system's behaviour in circumstances that are beyond the regular range of its intended use. A normal requirements specification cannot be used directly to enable negative testing since it only describes what a system is anticipated to behave in particular circumstances rather than addressing unexpected scenarios the system may meet and how to handle them. Thus, it is often left to a tester's skill and imagination to identify all the unusual and unexpected circumstances that may be of interest and to offer test cases verifying how the system will respond to them [4].

A mutation testing-based solution to the issue of supplying negative test cases for software system testing is discussed in this work. Mutation testing was initially used to evaluate the quality of sets of positive test cases by examining their capacity to identify defective copies of a tested programme (referred to as mutants) created from the program's source code by making minor changes to it. However, because it offers a generic framework for creating certain "faulty" artefacts, it may also be used to change positive test cases in order to create negative test cases. This method of obtaining test cases can reflect a wide range of unforeseen usage situations for a system and improve testing. the idea of using changing test cases as a support for model assessment rather than a programme. In this study, the idea is expanded upon, formalized, and applied to object-oriented software systems. It presents a general architecture for supplying and using the test cases and explains formally the rules (known as mutation operators) governing the production of negative test cases for such systems [5].

Negative testing tries to expose a system's weak points and determine whether it can handle unforeseen circumstances effectively or at the very least recover gracefully in the event of a failure. Although the problem has not received much attention specifically, several general testing methodologies offer suggestions for how to handle it. The most well-known application of these strategies is equivalence partitioning. This technique works by breaking down a system's input domain into a number of equivalence classes, including classes that represent erroneous values, and then choosing one value from each class to serve as the basis for test cases [6].

CONCLUSION

Hence, Negative test cases are those that are defined based on invalid classes. However, these methods only focus on the input data; they therefore only address one class of invalidating typical system usage scenarios. Stress testing is another method of testing that is strongly related to negative testing. Stress testing

Correspondence to: Jaweed Abdul, Department of Pharmaceutical Chemistry, Jahangirnagar University, Savar Union 1342, Bangladesh, E-mail: jaweedabdul@gmail.com

Received: 14-Jul-2022, Manuscript No. LDAME-22-18757; **Editor assigned:** 18-Jul-2022, PreQC No. LDAME-22-18757 (PQ); **Reviewed:** 01-Aug-2022, QC No. LDAME-22-18757; **Revised:** 08-Aug-2022, Manuscript No. LDAME-22-18757 (R); **Published:** 16-Aug-2022, DOI: 10.35248/2385-5495.22.8.016.

Citation: Abdul J (2022) Approaches to Negative Testing Using Mutation Testing. Adv Med Ethics J. 8:016.

focuses on examining how a system responds to extreme circumstances; as a result, typical test cases supplied to achieve this goal attempt to overburden or deplete the system of resources. Such methods, however, do not account for less dire circumstances that can arise from subtler adjustments to the way a system is used.

REFERENCES

1. Strug J. Mutation testing approach to evaluation of design models. Key Eng Mater. 2014;572:543-546.
2. Reid SC. An empirical analysis of equivalence partitioning, boundary value analysis and random testing. IEEE Comput Soc Fourth International Software Metrics Symposium. 1997:64-73.
3. Zhang J, Cheung SC. Automated test case generation for the stress testing of multimedia systems. Software Pract Exper. 2002;32(15): 1411-1435.
4. Briand LC, Labiche Y, Shousha M. Stress testing real-time systems with genetic algorithms. Evol Comput.2005:1021-1028.
5. Barna C, Litoiu M, Ghanbari H. Model-based performance testing: NIER track. 33rd Int Conf Autom Softw Eng. 2011:872-875.
6. Jia Y, Harman M. An analysis and survey of the development of mutation testing. IEEE Trans Softw Eng. 2010;37(5):649-678.