International Journal of Open Source Software and Processes
Volume 12 • Issue 1 • January-March 2021

Optimization of Test Cases in Object-Oriented Systems Using Fractional-SMO

Satya Sobhan Panigrahi, KIIT University (Deemed), Bhubaneswar, India
Ajay Kumar Jena, KIIT University (Deemed), Bhubaneswar, India

ABSTRACT

This paper introduces the technique to select the test cases from the unified modeling language (UML) behavioral diagram. The UML behavioral diagram describes the boundary, structure, and behavior of the system that is fed as input for generating the graph. The graph is constructed by assigning the weights, nodes, and edges. Then, test case sequences are created from the graph with minimal fitness value. Then, the optimal sequences are selected from the proposed fractional-spider monkey optimization (fractional-SMO). The developed fractional-SMO is designed by integrating fractional calculus and SMO. Thus, the efficient test cases are selected based on the optimization algorithm that uses fitness parameters, like coverage and fault. Simulations are performed via five synthetic UML diagrams taken from the dataset. The performance of the proposed technique is computed using coverage and the number of test cases. The maximal coverage of 49 and the minimal number of test cases as 2,562 indicate the superiority of the proposed technique.

KEYWORDS

Fractional Calculus, Graph Construction, Spider Monkey Optimization, Test Case Generation, UML Diagram

1. INTRODUCTION

Software engineering aims to design, analyze, deploy, implement, and maintain the software. The development of monolithic is not very effective for the development of the modern system. Therefore, several phases are established in the software development phase, termed the Software Development Life Cycle (SDLC). In SDLC, the testing phase is very necessary for several project resources. The software testing aims to identify the program structure faults and errors. Software is available in everywhere of our society like educational field medical field (Vola, et al., 2020; Carrado, et al., 2020), business, communication and almost in everywhere the software is available. Here, the software testing is carried out effectively and professionally (Kyaw, *et al.,* 2015) for controlling the system to be tested (Srivastava, et al., 2012; Muhammad, 2008). Test cases are the measure for collecting the required data input to produce the desired output (Jena, *et al.,* 2014). Besides, the test cases are generated to find the test cases and the critical domain requirements. The development of the test case accumulates the requirement specification of the program accurately (Hessel & Anders, 2006). One of the testing approaches is model-driven testing, which represents the behavioral approach for encoding system

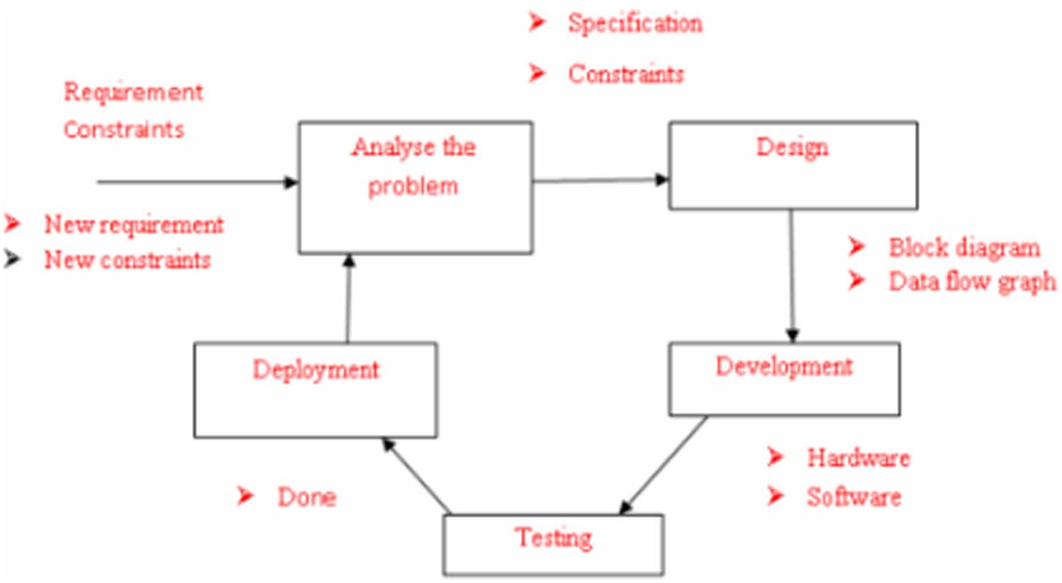
DOI: 10.4018/IJOSSP.2021010103

behavior with certain conditions (Priya, *et al.,* 2013). Automated testing is the model, in which the system generates the test cases automatically (Panigrahi, *et al.,* 2020). In the software system, the test scenarios are created by analyzing neither the activity diagram nor their corresponding source code. The UML-enabled scenario is generated based on the subsequent edge for designing the software. It is important for the test case design to enhance the quality of software, and to mitigate cost (Tripathy, *et al.,* 2013). Also, the UML indicates the structure and behavior of a higher-level system (Sahoo, *et al.,* 2017). Furthermore, the UML is the modeling language, which becomes very popular in several software engineering fields, like traceability analysis, testing, maintenance, and so on (Panigrahi, *et al.,* 2018) . In an object-based oriented model, the UML activity diagram is capable of designing an object, as it is suitable for illustrating the control flow (Nayak, *et al.,* 2011).

In the software development, maintenance, and testing of object-based oriented systems are corresponding to the execution in the system. Figure 1 shows the design and development of Object Oriented system for test case generation. An object-oriented operating system is an operating system that uses object-oriented design principles. An object-oriented operating system is in contrast to an object-oriented user interface or programming framework, which can be run as a non-object-oriented operating system like DOS or Unix. In an object-oriented system, all the data are represented as discrete objects where the user and other objects may interact. Each object has data as well as information about the executable file needed to interpret that data. An object-oriented system allows the user to focus completely on tasks rather than tools. Examples of object-oriented programming languages are C++ and Smalltalk.

The design test generation process will make it easier for the software engineer to assess configuration issues. Automated test case generation comprises of executing test cases automatically. To test the case, recognizing all the feasible start to the terminating paths in the activity diagram is very necessary for covering all the activities with their related control constructs. In this case, the control flow is performed based on looping, decision, concurrency, synchronization, and so on, to execute the sequence of several activities (Arora, *et al.,* 2017). In the last few years, UML-enabled testing has been utilized by several researchers for producing the test cases in the development cycle (Rhmann, et al., 2016; Pradhan, et al., 2019; Lam, et al., 2012; Jena, et al., 2015a; Jena, et al., 2015b; Jena, et al., 2015c; Panigrahi, et al., 2018). When prioritization approaches based on code are examined by

Figure 1. Design and development of Object Oriented system for test case generation


Specification
Constraints
Requirement
Constraints
Analyse the problem
Design
New requirement
New constraints
Block diagram
Data flow graph
Deployment
Development
Hardware
Software
Done
Testing