International Journal of Open Source Software and Processes
Volume 13 • Issue 1

Spider Monkey Particle Swarm Optimization (SMPSO) With Coverage Criteria for Optimal Test Case Generation in Object-Oriented Systems

Satya Sobhan Panigrahi, School of Computer Engineering, Kalinga Institute of Industrial Technology, Odisha, India*
Ajay Kumar Jena, School of Computer Engineering, Kalinga Institute of Industrial Technology, Odisha, India

ABSTRACT

System modeling in software engineering describes the procedure of creating a demonstration of a real system in a conceptual way to know the system behavior. The software testing promotes reusing the model for the function of testing, and this accelerates test case generation development. The test case generation verifies the reliability of system through enhanced test coverage. Test cases are the set of variables or conditions that define the quality of product and level of correctness. This research aims to develop a method named spider monkey particle swarm optimization (SMPSO) algorithm to generate test case using UML diagram. Accordingly, the proposed algorithm effectively generates the optimal test case by UML diagram through the construction of control graph. The proposed method achieved the coverage as 76 and generates the number of test cases as 82413.

KEYWORDS

Particle Swarm Optimization (PSO), Software Engineering, Software Testing, Test Case Generation, Unified Modeling Language (UML Diagram)

1. INTRODUCTION

In the software development strategy, the test cases can be generated with UML diagrams. Based on the functionality, the problems and the bugs are identified earlier to save time. The test cases are generated such that irrelevant test cases are reduced using the framework of objects and classes. Various traditional methods are used on various applications, like Java, where the tests are effectively generated and executed (Dhir, 2012). Testing is the procedure of performing the source code by varying inputs under different scenarios and is various contexts to ensure whether the actual performance of the system met the expected behavior in all the scenarios and contexts (Zamani &Hemmati, 2021). Software testing is the common and widely used method for verifying the software system and is generally used in recent decades in various applications (Savyanavar and Ghorpade, 2019; Thakur,

DOI: 10.4018/IJOSSP.300750 *Corresponding Author

Copyright © 2022, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

2018; Kumar, et al., 2020). However, software testing is a complex activity (Bertolino, 2007; Paiva *et al.,* 2020). The software becomes more complex and so testing the actual behavior manually is not feasible in the system (Moreira *et al.,* 2017; Paiva *et al.,* 2020). Hence, automatic software test is the solution to handle the restriction of time and the computational complexity. However, different software testing tools are available such that they are classified based on their criteria (Poston & Sexton, 1992). With regards to the degree of automation, most of the tools only automate test execution, whereas the others automate the generation of the test case, which are not required to be printed by the testers manually, but they are still needed to execute it physically(Paiva *et al.,* 2020).

The UML testing offers the concepts that consider both systematic and pragmatic development of the concise test models and test specifications(Baker *et al.,* 2004). Various concepts are utilized in the UML including test architecture, testing time, and test behavior. The test architecture shows the structural properties of test system such that it includes test context, the system under test, components of the test and the scheduler. Accordingly, the behavior of test shows the evaluation and actions that are required to define the goal of test. The behaviors approaches are utilized to indicate the behavior of the scheme. Some instances of the behavior diagrams is state machines, interaction diagrams, and activity diagrams such that they are used to define test control, test stimuli, actions and coordinations. The test time concept is used to complete the concept required for modeling the test and the timing concepts contain timers and time zones in UML for controlling and manipulating the behavior of the test and to integrate the components within the distributed system (Dhir, 2012). In the past decades, UML based testing model faces some challenges in the complex and large software industry system. However, the integration of UML specification in the software development system changed the development way and also it changes the testing process. Moreover, it causes shortening errors in the design phase. In general, executing the object-oriented program can start with the instantiation of thingsand thereafter message is passed to one of the objects or itself to process the calculation. However, messages are used to specify the dynamic observation of structure in the UML behavioral model. With testing perspective, the UML transition sequence from various artifacts offers essential information regarding the messages, like receiver or sender object, guard conditions and associated parameters (Shirole & Kumar, 2013).

To automatically generate the test cases, the tools utilize the software models and certain coverage factors for guiding them (Jena, et al., 2014; Jena, et al., 2015a; Jena, et al., 2015b). In most the time, such schemes do not exist and may be outdated and it is more complex to revise or construct them. However, in this case, generating a test case can resort to some external artifacts (Paiva *et al.,* 2020). Various solutions are developed in recent decades for automating the test case generation process among which the search base method is more effective in different domains and systems (Ali *et al.,* 2009; Harman & Jones, 2001; McMinn, 2004; Zamani &Hemmati, 2021). On the other hand, various publicly available tools, like EvoSuite are considered testing the Java programs such that they are regularly maintained and increased (Fraser & Arcuri, 2011; Zamani & Hemmati, 2021). The generation of the test case (Jena, et al., 2015c; Panigrahi, et al., 2018) has gained an important role in the testing mechanism. The process of testing manually exhausts 40 to 70% of time as well as cost in the development of software process (Maragathavalli, 2011). Different automatic methods (Panigrahi and Jena, 2020;Panigrahi, et al., 2021;Panigrahi and Jena, 2021) are designed for generating test cases to solve the issues of time expenditure in the physical testing procedure. The search-based testing (SBT) is the method employed to create test caseswithout human intervention. The software testing by metaheuristic or searching methods is called search-based testing (McMinn, 2011). Different metaheuristic methods, namelyPSO, Tabu search, genetic algorithm (GA)(Swamy, et al., 2013; George, et al., 2012), cuckoo search,ant colony optimization, simulated annealing and hill climbing are successfully used in the generation of test cases (kulkarni& Senthil, 2019; Srinivas &Santhirani, 2020; Sahoo & Ray, 2020).

This research is designed to model a new optimization algorithm named SMPSO algorithm to create optimal test cases from the Petri Net Markup Language (PNML) data. The proposed methods

18 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/article/spider-monkey-particle-swarm-optimization-smpso-with-coverage-criteria-for-optimal-test-case-generation-in-object-oriented-systems/300750

Related Content

Dynamic Key-Based and Context-Aware Internet of Things Authentication Approach
Mihir Mehtaand Kajal Patel (2022). *International Journal of Open Source Software and Processes (pp. 1-15).*
www.irma-international.org/article/dynamic-key-based-and-context-aware-internet-of-things-authentication-approach/310939

Open Source Urban Governance: Crowdsourcing, Neogeography, VGI, and Citizen Science
Carlos Nunes Silva (2015). *Open Source Technology: Concepts, Methodologies, Tools, and Applications (pp. 209-226).*
www.irma-international.org/chapter/open-source-urban-governance/120916

Code Ownership
Barbara Russo, Marco Scotto, Alberto Sillittiand Giancarlo Succi (2010). *Agile Technologies in Open Source Development (pp. 124-132).*
www.irma-international.org/chapter/code-ownership/36500

Helping to Bridge the Digital Divide with Free Software and Services
Jason G. Caudill (2010). *International Journal of Open Source Software and Processes (pp. 13-27).*
www.irma-international.org/article/helping-bridge-digital-divide-free/53875

Ensemble Techniques-Based Software Fault Prediction in an Open-Source Project
Wasiur Rhmannand Gufran Ahmad Ansari (2020). *International Journal of Open Source Software and Processes (pp. 33-48).*
www.irma-international.org/article/ensemble-techniques-based-software-fault-prediction-in-an-open-source-project/260972