Chapter 55
Framework for Reusable Test Case Generation in Software Systems Testing

Kamalendu Pal
https://orcid.org/0000-0001-7158-6481
City, University of London, UK

ABSTRACT

Agile methodologies have become the preferred choice for modern software development. These methods focus on iterative and incremental development, where both requirements and solutions develop through collaboration among cross-functional software development teams. The success of a software system is based on the quality result of each stage of development with proper test practice. A software test ontology should represent the required software test knowledge in the context of the software tester. Reusing test cases is an effective way to improve the testing of software. The workload of a software tester for test-case generation can be improved, previous software testing experience can be shared, and test efficiency can be increased by automating software testing. In this chapter, the authors introduce a software testing framework (STF) that uses rule-based reasoning (RBR), case-based reasoning (CBR), and ontology-based semantic similarity assessment to retrieve the test cases from the case library. Finally, experimental results are used to illustrate some of the features of the framework.

INTRODUCTION

The strategic importance of software has long been understood by professionals and policymakers around the world (Dutta, Wassenhove & Kulandaiswamy, 1998). Software has become an indispensable part of every industry, from mobile phone manufacturers to the safe landing of spaceships. The advancement of software design and development is always an open topic for researchers to address complex systems with numerous domain-specific requirements. The success of a system is based on the quality result at every stage of development, and therefore, superior software development is an essential element of success. These quality-enhancing features become more important as the transition from hardware to software-

DOI: 10.4018/978-1-6684-3702-5.ch055

enabled products accelerates. Today's technological innovation in the software industry is similar to the early 1970s, when digital electronics began to replace the mechanical and analogue technologies that underpinned the products of attractive desktop calculators to black-and-white televisions.

The landscape of the top software dependent product and service companies is changing rapidly. The value is evolving fast as hardware features are mostly standardized and differentiate software between high and low end products. At the same time, more miniaturized computing power provides the value of embedded software in products is expected to continue its market demand. In fact, software enables an estimated *high percentage* of automotive product innovations, from entertainment to fatal accident avoidance systems (Charette, 2005; Charette, 2009). These new-generation, software-based touch-and-feel-sensitive systems are highly dependent on human-computer interactions. Computer interfaces are becoming more and more sensitive and demanding as the number of products increases, from biometrically detectable automotive unlocking systems to sophisticated automotive dashboards that design and use *intelligent software* displays. As software-based user interactions become the operational norm, software-driven automation of the design and development of these software systems promise a new degree of quality while reducing production costs. A company with consistently high-performing software has less downtime in operation and develops products with fewer disruptions that impact the end-user experience.

The automotive manufacturing industry is witnessing a new era of software-driven innovative products and processes. Particularly, using the Internet of Things (IoT) technologies and enhanced computing power in solving operational problems. For example, self-driven cars are slowly becoming a reality with high hope of commercialization for public transportation soon. The world would now have self-driven cars if the automotive industry could model very accurately the randomness of human-drivers and pedestrians on public roads. One solution to this problem would be restricted lanes for autonomous vehicles only. Self-driving cars can communicate and coordinate with each other that human drivers often fail to do. It should be noted that operational data analysis and error-free software systems are the main driving forces of these self-driven cars.

The road traffic and observation data of the self-driven car can reveal interesting patterns and correlations in the collected data set. The important mechanism for finding causal relationships is often guided by intelligent data analysis software controlled by artificial intelligent (AI) techniques. For example, algorithmic sorting, searching and data clustering are often used to analyze the observation data. In this way, the introduction of AI-based software applications improves human performance in using these automated systems.

Academics and practitioners (Holcombe, 2008) are pushing for innovations in software design and development based on artificial intelligence. At the same time, technological innovations and their applications are transforming information and communication technology (Prahalad & Mashelkar, 2010). The world of software design and development also focuses on technological innovation to enable efficient software design and development processes. True, innovative technologies and techniques are now shaping everything, from capturing software requirements to delivering software, and as new perspectives emerge, fully automated customer service in software development processes are on the horizon. The technologies that enable global software design and development, such as software development platforms and tremendous computing power, data storage, and innovative software testing techniques are rapidly advancing and becoming increasingly possible.

Software being developed in the industry requires careful operational planning and the coordination of required resources. The inception, design, implementation, testing and maintenance of a software