Clustering of Web Application and Testing of Asynchronous Communication

Sonali Pradhan, SOA University, Bhubaneswar, India
Mitrabinda Ray, Siksha O Anusandhan Deemed to be University, Bhubaneswar, India
Srikanta Patnaik, Siksha ‘O’ Anusandhan University, Bhubaneswa, India

ABSTRACT

In model-based testing, testing is done using Input Output Transition System (IOTS) or Finite State Machine (FSM). In synchronous testing, it is easy to know the proper sequence of input output of the system under testing, whereas, in asynchronous communication, it is tough to trace the sequences of inputs and outputs. A web application is an example of asynchronous processing, in which input/output queue management is a common issue for testing in asynchronous JavaScript and XML user interfaces. This article discusses asynchronous testing of web application by taking a Gmail account as a case study. In this article, the web application is partitioned into clusters by determining logical web pages that implement logical functions. The clusters of page objects are represented using finite state machines. From the FSMs, test scenarios are generated. The size of the test cases covering longer paths is decreased by making clusters of states. The authors conduct a queue analysis using a test architecture to investigate the behavior of the system under testing. Finally, the test cases are generated using the Selenium tool.

KEYWORDS

AJAX, Asynchronous Testing, FIFO, Input Output Transition System, Modelling Web Application, Non-FIFO Channel

INTRODUCTION

Testing has an important part of the software development process. However, the manual process of testing is extremely costly and error-prone. This leads to Model Based Testing (MBT), where the behavior of the System Under Test (SUT) is tested based on the functional aspect of the model. (Barnett, Grieskamp et al., 2003; Farchi, Hartman & Pinter, 2002; Holt, Briand & Torkar, 2014; Utting, Pretschner & Legeard, 2012). These MBT models are state-based. With the help of MBT tools, these models are either translated to Finite State Machine (FSM) or in the Input Output Transition System (IOTS). In synchronous communication, a single input output system is maintained. Synchronous testing is done using FSM models, and FSM (Holt, Briand & Torkar 2014; Utting, Pretschner & Legeard, 2012, Pradhan, Ray & Patnaik, 2019) has a finite number of states with transition between

DOI: 10.4018/IJACI.2019070103

Copyright © 2019, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
them. FSM can handle single input & output queues. However, IOTS handles multiple input and output queues. IOTS models are commonly used when next input can arrive even before current outputs are produced. IOTS usually models concurrent input/output behavior with transition systems. Like FSM, an IOTS (Hierons, 2012; Aho & Dahbura, 1991; Lynch & Tuttle, 1988) is also defined as a simple type of state machine with states and transitions between them. A single input represents FSM and single output queue whereas, multiple input and multiple outputs represent IOTS. FSM represents each single input output queue. Some examples of IOTS models are shown in Figure 3 & 4.

In the asynchronous process, the number of input and output messages are not in a sequential manner. Due to delay in output, the receiver receives more than one input messages. So, observed sequence of input and output may not be the trace of the system under test. It is difficult for the tester to know which output is for which input and it is very tedious to generate test cases synchronously. So, multiple IOTSs are maintained in asynchronous communication, which makes tedious and time-consuming to test. In synchronous processing, the observed sequence is the correct sequence of input and output of the system under test which is not the same in the case of asynchronous testing. The asynchronous channels also work with message passing system to communicate with the system under test. A widespread example is taken as a web service. Due to delay in the asynchronous channel, the system sometimes affects the testing, which is also cannot be observed by the tester. In the asynchronous channel, the capacity of the buffer is non-zero, messages may wait in the buffer before the receiver retrieves them. In asynchronous testing, two types of channels are tested- (i) First-in-First-out (FIFO) and (ii) non-FIFO channels through IOTS. In FIFO, it is a pair of <send, receive> messages. When there are two input messages are send the FIFO property violated. To introduce a model of the communications channel, the asynchronous nature of communications is taken into account to generate test cases. In the asynchronous channel, the tester observes that the output is delayed from the SUT and the input supplied by the tester is received later by the SUT (Hierons, 2012; Aho & Dahbura, 1991). Thus, a trace in which the sequence of inputs and outputs observed by the tester might not be the correct sequence of a trace of the SUT. Taking an example, where the trace has the sequences \( ?_i, !_o \), by sending input \( ?_j \) twice and then observes the output \( !_o \), it is quite possible that output \( !_o \) was produced by the SUT before any input was received and another possibility may be the output is the response of the first input.

If the communication channel is non-FIFO, the situation is deplorable. The inputs sent by the tester and the outputs produced by the SUT might be in a different order. In this situation, outputs generated by the SUT might also require reordering (Hierons, 2012; Aho & Dahbura, 1991). If the channel is unbounded, the communication channel has an infinite number of states. There is no natural way because there are no machines with unbounded memory. So user always wants a bounded, no magical way how the system can sustain an infinite number of requests and makes free from solving an overload problem. It is not possible to describe messages to be recorded in the global state in a non-FIFO system. In anon-FIFO system, either some degree of inhibition or piggybacking of control information is used in computation messages to capture out-of-sequence messages. Here, Figure 1 and Figure 2 clearly illustrate the difference between synchronous and asynchronous communication techniques.

The synchronous communication process is shown in Figure 1. It shows how a process has to be completed before another process is started. Thus, this leads to a stop and wait for approach, whereby the user has to wait until one request previously made has completed fully to another request to be processed.

Asynchronous communication between client and server is shown in Figure 2. In this figure, we find the asynchronous process, where the client-server communications are done using Asynchronous JavaScript and Xml (AJAX) application. The reader can refer to these papers (Garrett, 2005; Salva, 2009; Carteri, 2010; Stocco, 2017) for detailed information about AJAX.

In the web application, asynchronous communication is more appropriate. It includes queues to store the sequence of input and output events. Here, an IOTS implementation is tested
Bidder Selection in Public Procurement using a Fuzzy Decision Support System
www.igi-global.com/chapter/bidder-selection-in-public-procurement-using-a-fuzzy-decision-support-system/178455?camid=4v1a

Content-Based Image Classification and Retrieval: A Rule-based System Using Rough Sets Framework
www.igi-global.com/article/content-based-image-classification-retrieval/2422?camid=4v1a
A Semantic-Enabled Middleware for Citizen-Centric E-Government Services
[www.igi-global.com/chapter/semantic-enabled-middleware-citizen-centric/64377?camid=4v1a](www.igi-global.com/chapter/semantic-enabled-middleware-citizen-centric/64377?camid=4v1a)

Fuzzy based Quantum Genetic Algorithm for Project Team Formation
[www.igi-global.com/article/fuzzy-based-quantum-genetic-algorithm-for-project-team-formation/145776?camid=4v1a](www.igi-global.com/article/fuzzy-based-quantum-genetic-algorithm-for-project-team-formation/145776?camid=4v1a)