JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-12, May 2019 E-ISSN: 2347-2693

Test Case Automation Using Natural Language Processing

Saurabh Mahajann[1*], Mona Mulchandani[2], Samir Ajani[3]

[1,2,3]Department of Computer Science& Engineering, Jhulelal Institute of Technology, Nagpur, India

Corresponding Author: saurabhvmah@gmail.com

Available online at: www.ijcseonline.org

***Abstract*—** Software testing is an integral part of the software development life cycle. In order to achieve the testing on individual module or complete integration testing of the system a quality analyst makes test cases for it, covering the different scenarios against which validity of the working of the system is checked. A test case is nothing but a set of conditions and data with it under with it a quality analyst will test a system and check if working according to specification or not. More focus is made on automating the testing of the software, resulting in the creation of the several automation testing framework such as Selenium, Apache Jmeter etc. Similarly attention on creating the test cases is also made. As user stories are written in the natural language, natural language processing can be a way of extracting information from it. Conformiq provides the same functionality. In agile development methodology such as extreme programming such automation is really advantageous which provided quite flexibility and adaptive to change provided.

***Keywords*—** Software Testing, Automation,Test Cases, Selenium, Conformiq, Agile Methodology, Natural Language Processing, Test Driven Development.

I. Introduction

Generally, typical IT Company is having considerable portion of their budget in testing related activities in their whole project.

In final user acceptance testing software system will be validated by the product owner before acceptance. The success of the user acceptance testing process thoroughly depends upon the number of errors found and rectified before releasing the system and hence ultimately depends upon the test cases written for the testing of the system.

A test case can be used by both quality analyst and software developer. QA can use it for validating where a developer will use the test case for writing the unit test case for the individual module. A test driven development involves creating of the test cases first before even starting actual development.

In the long journey of the information technology enhancement various methods have been proposed for generating test cases. A test case is nothing but a set of conditions and data with it under with it a quality analyst will test a system and check if working according to specification or not.

Test cases can be mapped directly to, and derived from use cases. One of the ways of deriving the test case is to make it from software system requirements and specification.

The main reason and advantage behind generating test cases based on software requirement and specification is that they can be created in very early phase of the software development life cycle and hence ready for both QA and software developer .

With this another thing can be achieved is that software developer might while developing may find certain inconsistency or ambiguity in the requirement while designing the unit test for the system, so such errors, ambiguity and inconsistency can be removed in early phase and hence reducing the extra effort QA might require while validating the requirement.

In test driven development a test case is what developer needs to consider without actually starting with the development. In such case test cases should be constructed covering all the scenarios, missing in a single scenario may cause a software system not work in that specific condition as code is directly written according to the test cases.

In the last, the entire system undergoes testing in the context of a Functional Requirement Specification (FRS) and System Requirement Specification (SRS), where it tests the design and its behavior.

II. Related Work

Since the test cases are fundamental blocks of the overall testing of the product or even unit testing made by the developer, there are many approaches till now have been made on automating the generation of the test cases and extensive research and study have been performed on the same. Different approaches used different paradigm to achieve.

The aim was same, anyhow automate the process and reduce human prone error, i.e. missing any important criteria while testing which in future might cause a big impact and even may cause software system to collapse down.

W.T. Tsai, D. Volovik , T.F. Keefe proposed a same on the basis of relational algebra, titled “Automated test case generation for programs specified by relational algebra queries”[]. Here the constrain was to write the specification in the form of relational algebraic expressions only and hence limiting the scope. It’s not always feasible to represent the requirement in algebraic expressions.

Some approaches focused on diagrammatic models. Valdivino Santiago, Ana Silvia Martins Do Amaral, N. L. Vijaykumar, Maria De Fatima Mattiello-francisco put forth a approach titled “A Practical Approach for Automated Test Case Generation using Statecharts”[]. In this implementation software specifications and requirements modeled to the states charts. So here normal user story was not considered directly. States charts are then converted into the transitions diagram.

Other approach includes the one using domain specific model. Arne-Michael Torsel proposed *Automated Test Case Generation for Web Applications from a Domain Specific Model* [] , here black box automation testing was addressed.

III. METHODOLOGY

In related works different approaches for the constructing test case were discussed. Survey shows there was always some constrain in which their methodology will work.

But in our day to day practices all the software requirements and user story are represented in natural language, the one which is understood by the human and not the computers directly. User stories are written in such a way that developer and quality analyst can easily understand it and hence can started working on it as no complexity in the stories written in simple language.

To remove constrains for specifying software requirement and user story in particular format, and keep it in the natural language, the only approach came to rescue is natural language processing which can extract and understand the user requirement from the user story which also describes different use cases for the requirement and hence test cases can be constructed on the basis of that my mapping the use cases.

Natural Language Processing:
Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and is a component of artificial intelligence (AI).
The development applications which involves Natural language processing is challenging as computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured, or using a limited number of clearly enunciated voice commands. However, human speech is not always precise, it is ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

The main aim of NLP is to analyzing and understanding and base on gathered information it generates a language in both written and spoken form. NLP is nothing but the making the computer to understand how humans learn and uses language. The base of NLP has different area as follows:

- Computer science
- Linguistics
- Mathematics
- Robotics
- AI

The Other thing our test case generation in based on is software requirement and specification.
Software Requirement Specification

A software requirements specification (SRS) is nothing but a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is generally developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system and individual module.

The software system requirement specification document consistent of all necessary requirements required for project development. It provides how system will perform in real time environment, its interaction with user and what is the final outcome from the system.
A good SRS possesses following characteristics:

- Unambiguous
- Complete
- Verifiable
- Consistent
- Modifiable
- Traceable
- Usable during the Operation and Maintenance Phase

It includes following content:

1) Introduction
It includes a purpose of document, scope of product, intended audience of the product along with the document

overview, definitions, acronyms and abbreviations, references and acknowledgments.

2) Overall Description
It includes product perspective, product functionality, users and characteristics, operating environment, design and implementation constraints, user documentation, assumptions and dependencies.

3) Specific requirements
It includes External interface requirements, functional requirements, Public Relation Department (PRD) tables and behavior requirements.

4) Other Requirements
It includes appendix which may include Terminology or Glossary or Definitions list which should be in alphabetical order.

B. Functional requirement
Requirements form the basis of any system's foundation. For any product development life cycle determining the right requirement for system is very critical it should be well thought out ,clearly understood in order to not to be doomed to failure and hence emphasis on this phase should be given.
For a developer all the software requirement and specifications is divided into different user stories. In test driven development developer must generate test scenarios based on the user stories.

So the methodology will consists of the three modules, one consists of the input i.e. functional requirement which may include software requirement and specification and user stories, another being an automation engine consists of natural language processor in a core of it. Final will be output i.e. automatically generated test cases.

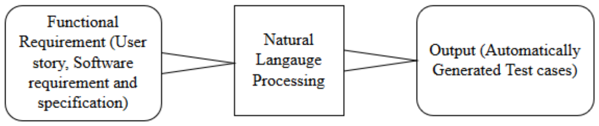
Functional Requirement (User story, Software requirement and specification)
Natural Langauge Processing
Output (Automatically Generated Test cases)

Fig 1: Proposed System Block Diagram

Fig 1 shows the brief about how methodology will go to work practically. NLP engine will be a core to all, which will extract the scenarios.

Test Driven Development:
Test-driven development is a recommended agile practice. TDD is a special software development technique integrates design with development process: first write unit tests, and then write the code to pass the tests. Test code writing and making test code passed can get enough feedback and make the right decisions, also reduce the cost and repair bugs.
Steps involved in TDD generally include:

- Add a test Run all tests and see if the new one fails
- Write some code
- Run tests
- Refactor Code
- Repeat

The reason behind TDD to mention here is that proposed survey system will make TDD even simpler.
Test Case Generation Techniques:
There are multiple techniques available for generating test cases:

Goal-oriented approach –
The purpose of the goal-oriented test case generation approach is to cover a particular section, statement or function of a user story or individual module. Hence here the execution path is not important and need not be considered, but testing the goal(user story) is the primary objective.

Random approach –
In the random approach, test cases are generated based on assumptions of errors and system faults.
Specification-based technique –
This model generates test cases based on the formal software system requirement specifications.

Source-code-based technique –
The source-code-based case generation approach follows a control flow path of individual module to be tested, and the test cases are generated accordingly. It tests the execution paths.

Sketch-diagram-based approach –
This follows generation of test cased on the basis of Unified Modelling Language (UML) diagram to formulate the test cases.

IV. Results and Discussion

From all the findings, it shows, different approaches were put forth for automating the test case generation. Software specification and requirement is converted into individual user stories. Natural language processing can be an effective tool to extract the test case scenarios from the user stories.
This will also help in individual module level to a developer while developing and individual module to validate it against the case scenarios extracted using natural language processing.

V. Conclusion and Future Scope

The survey shows natural language processing can be another approach for automating the generation of the test cases. As natural language processing is used there will not be any constrains on specifying software specification and requirement in particular format.

Automation will also reduce the human error where certain scenarios while creating test case by own can be missed. Future scope can be further study can be made on writing automated script where script will be generated automatically in order to execute those test cases.

REFERENCES

[1] Ahlam Ansari ; Mirza Baig Shagufta ; Ansari Sadaf Fatima ; Shaikh Tehreem , "Constructing Test cases using Natural Language Processing" , Published in: 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)
[2] Andrew Rendell, "Effective and Pragmatic Test Driven Development", Published in: Agile 2008 Conference
[3] "ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering", Published in: 1 Dec. 2011
[4] J.Zalewski, "How to write the SRS documentation, following IEEE Std. 830.", ISM 4331, J.Zalewski, September 2003.
[5] Roger Pressman , "Software Engineering A Practitioner's Approach 7th Edition" , McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, NewYork, NY 10020. Copyright © 2010 by The McGraw-Hill Companies, Inc.
[6] Hécio A. Soares and Raimundo S. Moura, "A methodology to guide writing Software Requirements Specification document", Departamento de Informática Instituto Federal do Piauí and Departamento de Computação Universidade Federal do Piauí, 2015 IEEE.
[7] Jai Gaur ; Akshita Goyal ; Tanupriya Choudhury ; Sai Sabitha, "A walk through of software testing techniques", 2017.
[8] Itir Karac ; Burak Turhan, "What Do We (Really) Know about Test-Driven Development?" , IEEE Software (Volume: 35 , Issue: 4 , July/August 2018)
[9] J.Zalewski, "How to write the SRS documentation, following IEEE Std. 830.", ISM 4331, J.Zalewski, September 2003.
[10] M. Costantino ; R.G. Morgan ; R.J. Collingham ; R. Carigliano , "Natural language processing and information extraction: qualitative analysis of financial news articles" , Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)
[11] Ron Patton , "Software engineering". 800 E. 96th St., Indianapolis, Indiana, 46240 USA.
[12] Shi Zhong ; Chen Liping ; Chen Tian-en, "Agile planning and development methods", Published in: 2011 3rd International Conference on Computer Research and Development. Shanghai, China.
[13] Suresh Thummalapenta ; Saurabh Sinha ; Nimit Singhania ; Satish Chandra, "Automating test automation", Published in: 2012 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland.
[14] W.T. Tsai ; D. Volovik ; T.F. Keefe, "Automated test case generation for programs specified by relational algebra queries"
[15] Roger Pressman , "Software Engineering A Practitioner's Approach 7thEdition" , McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, NewYork, NY 10020. Copyright © 2010 by The McGraw-Hill Companies, Inc.