Add Self-Learning Ability to NLP for Automatic Test Case Generation

SHRIKRUSHNA ZIRAPE[1], SHIVAM SHARMA[2], ILYAS HUSSAIN ALI[3], MANASI KUMBHAR[4], RENUKA NALAWADE[5], PROF. MANISH JANSARI[6]

[1, 2, 3, 4] *Pune Institute of Computer Technology, Pune, India*

[5] *Veritas Technologies (Principal SQA), Pune, India*

[6] *Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India*

Abstract- ***In Software testing, 40-70 percent of the testing process is spent on developing and designing test cases. It is tough for the untrained tester to generate all the test cases that cover every aspect of the criteria. By changing requirements frequently manual development becomes less valuable and it requires more time and effort. Rather than generating test cases manually, a tool can be used to generate test cases automatically based on user stories and scenarios, but the dictionary plays a very important role in this process. In this project we have used Natural language processing to generate dictionary in which it will find keywords in user stories or scenarios and create test cases accordingly. As this entire process is automated, it is very efficient in time perspective. This project provides a realistic solution for automatic dictionary generation which will be used for test case generation.***

Indexed Terms- ***Agile, Natural Language Processing, Dictionary***

I. INTRODUCTION

This thesis investigates if a vocabulary can be automatically created from user stories using natural language processing to create in an Agile workflow, the test cases that are required. The final goal is to produce functional test cases from business requirements. Agile uses user stories are used to record business requirements. Test scenario description and dictionary are two input factors needed for automatically producing test cases. Dictionary that is used for generating the test cases was formerly produced manually, but now using the NLP techniques we're aiming to produce the dictionary automatically.

The traditional waterfall model had a lot of limitations and was unable to keep up with rapidly growing and changing market. Thus, Agile software development methods was introduced to meet the requirements of market and business stakeholders. In agile software development methods, speed development is possible which helps in meeting business requirements. This is possible by having testing and development go hand in hand together from the very beginning. TestDriven Development (TDD) and Behaviour-Driven Development (BDD) are the most commonly followed techniques in Agile development methods. TDD was considered to be unstructured and hence BDD was developed to overcome these limitations. These methods were however introduced before the automated testing scenario and thus suffer from limitations such being not able to produce a full testing process.

The vocabulary includes commonly used system terms as well as testing techniques that might interfere. The user previously had to save keywords and test parameters pertinent to the implementation in the dictionary. Words from the lexicon are regularly used in user stories and related test phases. With the help of this we can cover a much broader range of test case scenarios. The software's edge cases are also tested. This is done using boundary and corner cases. For instance, the tester might want to check the parameter's limits by assigning the word "Quantity" values of zero, positive, and negative. A variety of test situations for different criteria are made available with the dictionary's assistance. The modification of the dictionary on the basis of functionality of the

system and the industry is also possible due to the fact that the user stories utilised will not be restricted to any particular area.

The dictionary can also be used as a technique for addressing errors in serious situations. The lexicon aids in increasing test coverage by diversifying the test methodologies. Existing natural language processing techniques are used to extract keywords from test case situations, filter them, and then their potential values are stored in a dictionary. The test cases are then created using these keywords.

II. LITERATURE SURVEY

Sr No.	Paper	Year	Summary	Limitations
1	Automatic Generation of Test Cases for Agile using Natural Language Processing	2017	Created a programm that automatically creates functional test cases from the free-form test scenario description using NLP approaches.	Increases the time by 7percent.
2	NLP-Based Requirements Formalization for Automatic Test Case Generation	2021	Created a tool based on NLP to generate requirement models	Cannot be used for complex textual data
3	Automatically Generating Tests from Natural Language Descriptions of Software Behavior.	2013	Uses natural language processing techniques, code information extraction and probabilistic matching.	Better techniques can be used to improve accuracy.
4	Automatic Generation of System Test Cases from Use Case Specifications.	2020	Uses Use Case Modelling for System Test generation approach .	Mostly only focuses on use case specification for generation and also the time taken is more.
5	Test Case Generation Using Activity Diagram and Sequence Diagram.	2012	Creates activity and sequence graph from the respective diagrams.	Cannot solve problem which are combination of activity and sequence diagram.
6	A Review of NLP Oriented Automated Test Case Generation Framework in Testing	2020	Literature review of how NLP is used in automated test case generations.	NLP in automated testing can increase performance but needs to work on accuracy.

TABLE I
LITERATURE SURVEY

III. SYSTEM AND FUNCTIONAL REQUIREMENTS

This section covers the software, functional and non functional requirements of the project.

A. Software Requirements

We have considered the following softwares in the course of the project development:

- Java
- Python

B. Libraries

- Numpy
- Pandas
- NLTK
- PyTorch
- TensorFlow
- SciPy
- Spacy

C. Functional Requirements

The following are the functional requirements for the proposed system:

1) SRS document from user for test case generation

The interested candidate should be able to upload SRS document from the User Interface from which the test cases will be generated. SRS document contains all specification and requirements required for test case generation

2) NLP Parsing

The SRS document must then be checked for proper formatting and successful NLP parsing should be performed using methods like POS tagging, syntactic parsing etc for further processing.

3) Convert dependencies to frames

In this step the POS tagging splits the keywords into verbs and adjectives and appends to dictionary as bigram.

4) UML diagram creation

The SRS document will pass through the NLP parser the operations like lemmatization and POS tagging will be applied on it and after that various uml diagrams will be generated like activity diagram, use case diagram.

5) Activity Graph traversing

The acitvity graph shows the relation and sequence of steps required to generate test case it forms relations between different components of the project so that the generated test case can cover the whole project.

D. Non-functional requirements Performance Requirement

1) User satisfaction:

The UI should be usable for any new user and it should be developed by considering all the requirements of the user.

2) Concise Result:

The application should generate all test cases as per mentioned in the SRS document

3) Accuracy:

The generated test cases should be correct and their accuracy should be high so that we cannot leave any test scenario

4) Application Availability:

Application availability is a measure used to evaluate whether an application is functioning properly and can be used to meet individual or organizational needs

Safety and Security Requirements

1) Improper or incomplete SRS:

The formatting and parsing of SRS to generate dictionary is the most important function and it is essential that the SRS submitted is verified and free of errors to generate proper dictionary.

2) Generating precise dictionary:

During NLP parsing we apply POS Tagging to separate adjective and verbs and care needs to be taken while appending verbs as bigram to dictionary.

3) Security:

SRS is a very critical document during software development life cycle and it is important to ensure that the system on which the SRS is taken as input and parsed is secure.

Software Quality Attributes

1) Availability

The system should be operationl and running and after receiving the first SRS it should be able to take second SRS after generating test cases for first one

2) Maintainability

The test cases should be generated for all modules of the project specified in the SRS document.

3) Usability

The input data which is large size should be handled by the system effectively.

4) Reliability

The software system Reliability is nothing but the executing probability of a function that has specific requirements, specific input, and a fixed number of input conditions in the specified time interval(The hardware and input not contains any errors)

5) Robustness

The impact of operational mistakes, incorrect input data, and hardware failures is reduced by robustness.

IV. RELATED WORKS

A. Different types of techniques using NLP

Designing and generating test cases is a tedious manual process that requires 40-70 percent of the software test life cycle. Also, often these are designed by inexperienced testers and does not cover all requirements of the project. Projects have now shifted from traditional waterfall models to Agile model. Here, the business requirements are captured in the form of user stories. The aim is automate the generation of test cases using natural language processing (NLP).

We make use of NLP techniques to capture these user stories in natural language format followed by transforming the natural language into computational models using UML. We develop a tool to generate test cases automatically by creating UML diagrams.

in [2] major focus is on how to create specification models from the functional requirements that is given in natural language. A technique using NLP is proposed to generate test cases automatically. This method tries to automate the creation of model from the given requirements by making use of various algorithms. It is not restricted to any specific domain or format and also covers a wide range of requirements formulation.

[2] also generates requirement models for battery charging approval system for evaluation producing correct and complete artifacts to a high degree which are used to create sequence diagrams to be able to finally generate abstract test cases.

[3] tries to describe a prototype tool, kirby, that can automatically translate natural language behavioural descriptions into exe test files. Behaviour-Driven Development (BDD) is an emerging agile technique that is built on the already established Test-Driven Development (TDD). TDD is an iterative process and here before writing the implementation code, the test code is written thereby introducing testability during the software design itself. Despite many advantages, TDD also has some drawbacks as it requires understanding of “where to start, what to test and what not to test, how much to test in one go, what to call their tests, and how to understand why a test fails”.

Thus, BDD was developed to overcome these limitations of TDD as it expresses the behaviour of system in natural language thereby improving communication among stakeholders by helping them understand the behaviours. The one limitation however of BDD is that programmers are still required to write program steps that corresponds to these behaviours/scenarios to translate into executable software tests and this can be referred to as ”glue code”. [3] proposes that instead of programmers having to write ”glue code” manually, the natural language scenarios can be converted to executable software test files automatically.

[3] makes use of a tool called kirby which generates executable software tests from the natural language structured scenarios using natural language processing techniques. A small collection of 12 BDD scenarios are compiled and ran through Kirby to evaluate and assess the performance as well as accuracy thereby showing the feasibility of one technique for accomplishing the task. Currently, this prototype has not undergone significant evaluation and thus as part of future work, a larger collection of existing BDD scenarios needs to be compiled for truly evaluating its effectiveness and performance.

B. Activity and Sequence Diagrams

Activity and Sequence Diagram Creating System graphs by integrating activity graph and sequence graph. Firstly they convert sequence diagram to sequence graph and activity diagram to activity graph and then integrate together to system graph and then use this system graph to further generate test cases. Once SYG is generated , it is traversed using DFS and used TCG-SYG which automatically traverses. It travels all the possible paths from node to node, checks for pre condition, selects the test case, checks the input with pre condtion and then post condition is checked and then the test case is added. In activity diagram conditional statement is requied for having the possibility of multiple paths and give optimal solution, in sequence diagram all cases should be considered as a result it will cover all the possibilities. It solves the problem of concurrent

exevution which can lead to state explosion problem. Since they use DFS thes cases obtained are exhaustive and optimum. But the limitation faced using this method was problems related to combination of activity and sequence diagram remained unsolved.

V. ADVANTAGES

There are a number of advantages some of which are

1) Maximum coverage of paths and data.
2) Reduces the difficulties to tester with no prior knowledge of the working of the system for designing the test cases.
3) Save time and money.
4) Improve quality of testing with better test coverage.
5) Easier maintenance and reuse of test cases if customer changes the requirements.

VI. LIMITATIONS

1) The input user stories' format is constrained.
2) The Description for the test case and the input user stories determine the significance of the test cases.
3) The performance of the Test Cases generated using this tool is dependent on how the user knows grammar and punctuation. The user can his Grammatical skills while defining the user story and test scenario.

CONCLUSION AND FUTURE WORK

To save time and effort spent on testing, automatic test case generation has recently been a study issue. Making the testing process less influenced by people has been the subject of several solutions and methodologies. A win-win situation is achieved when automated testing and natural language processing are coupled.

Test cases will be produced with a great deal less work and time, and they will be of higher quality. One of the crucial inputs for the programme that generates test cases is a dictionary. Dictionary will assist in producing test cases of higher quality and with greater test coverage of the requirements. The process of manually creating a dictionary takes a lot of time, hence automation is required.

REFERENCES

[1] Prerana Pradeepkumar Rane "Automatic Generation of Test Cases for Agile using Natural Language Processing" - March 14, 2017 Blacksburg, Virginia.

[2] Chunhui Wang, Fabrizio Pastore, Arda Goknil, and Lionel C. Briand - "Automatic Generation of Acceptance Test Cases from Use Case Specifications: an NLP-based Approach" - May 2020.

[3] Grootendorst, M. "KeyBERT: minimal keyword extraction with BERT, v0.1.3." (2020).

[4] Shweta Ganiger; K.M.M. Rajashekharaiah "Comparative Study on Keyword Extraction Algorithms for Single Extractive Document" 2018 Second International Conference ICICCS.

[5] Mohsin Irshad, Ricardo Britto, KaiPetersen Adapting Behavior Driven Development (BDD) for large-scale software systems - Elsevier 14 May 2020.

[6] Dipti Belsare, Dr. Manasi Bhate "A Review of NLP Oriented Automated Test Case Generation Framework in Testing" - International Journal of Future Generation Communication and Networking Vol. 13 - 2020.

[7] Robin Gropler1 , Viju Sudhi1 , Emilio Jos¨ e Calleja Garc´ ´ıa2 and Andre Bergmann - "NLP-Based Requirements Formalization for Automatic Test Case Generation" - AKKA Germany GmbH, 80807 Munchen,¨ Germany - 2021.

[8] Boghdady, P. , Badr, N. L., Hashem, M. A., Tolba, M. F., "An enhanced technique for generating hybrid coverage test cases using activity diagrams" Informatics and Systems.

[9] Ranjita Kumari, Vikas Panthi, Prafulla Kumar Behera,"Generation of test cases using Activity Diagram" International Journal of Computer Science and Informatics, ISSN (PRINT): 2231 – 5292, Volume- 3, Issue-2, 2018.