IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 09, Issue 5 (May. 2019), ||S (XI) || PP 42-45

Automating Test Case Creation Using Natural Language Processing

Saurabh Mahajann[1], Ms. Mona Mulchandani[2], Mr. Samir Ajani[3]
[1]*Department of Computer Science and Engineering, Jhulelal Institute of Technology,RTMNU, Nagpur*
[2]*HOD, Department of Computer Science and Engineering, Jhulelal Institute of Technology,RTMNU, Nagpur*
[3]*Assistant Professor, Department of Computer Science and Engineering, Jhulelal Institute of Technology,RTMNU, Nagpur*

Abstract : *In order to point out the defects and errors that were made during the development phases, testing of the any application program is necessary. It's really important to ensure that the application should not result into any failures because it may be very expensive in the future or in the later phases of the development cycle. To test the behavior of the system Quality Analyst constructs a test case on the basis of user story provided or from the Software requirement specification. Similarly a software developer needs to construct the test cases while writing the unit tests. In test driven development test cases are required even before writing a single line of code. To save the time this needs to be automated. Current implementation makes use of the Natural language processing to achieve the same. Implemented system makes direct use of Natural language processing hence there is no restriction on the format for writing software requirement specification or writing the user stories. This automation overall removes the overhead associated change in the requirement as no manual efforts are required further and hence current implementation supports truly agile software methodology.*

Keywords : *Natural Language Processing, Test Driven Development, Test Case Generation, Agile Software Development, Software system specification, User Stories.*

I. Introduction

A testing of the software system is one of the important tasks in the software development process. Quality analyst on the basis of the software requirement specification or user stories constructs test cases and tests the system against it. This is quite time consuming. Also small change in the requirement may affect the whole testing scenarios and may result in creating the test cases again based on the change and hence the extra overhead is associated with it. Similarly developer when working on the particular user story also need to construct the test cases for performing the unit testing on it or when following the test driven development has to think about the test cases even before writing the single line of the code and which is time consuming and might miss the important condition if the user story is quite large. To overcome these difficulties the need of automating the same came into the focus. This led to automate the existing system and several approaches have been put forth to achieve the automation.

However each of them having the drawbacks associated it. Some of the implementation requires the requirements specification to be hard coded into the particular format, e.g. in relation algebraic expression [] which is not user friendly. Another implementation was based on the states charts [] and hence states charts are needed first which is not feasible. Some implementations were domain specific [] where system need to be trained first in that particular keeping the limited scope. Hence the system with some intelligence, which can read a human readable language, ignores the hardcoding is required.

To overcome the limited or restricted scope of the existing systems current implementation has been developed which not only overcomes these limitation of scope but also provided the lots of extending ability which can be used in the any domain. Proposed implementation makes use of the natural language processing which is acted upon the Software requirement specification or more at granular level on the individual user stories which might be dependent or independent with respect to each other. With the help of the Natural language processing, user story or SRS has been analyzed.

The main purpose of the implemented system is to reduce the overhead associated with construction of the test cases, implementation has been linked with the bug tracking and project management tool and hence keeping the whole view clear for the both developer and quality analyst and other members involved in the whole project.

In Short, Natural language processing (NLP) is the ability of a computer system program to understand humanly understandable language as it is spoken and is a component of artificial intelligence (AI). The main challenge here is the diversity present in the human speeches. There is always a chance that the language that human being speech is ambiguous and not the strict to the context. While implementing the approach, proper care while processing the human readable language has been taken.

This approach works at more granular level along with the Software requirement specification that's mean it user story level. In project management, User stories are noting an informal, human readable means natural language description of one or more features of a software system. User stories are often written in the perception of an end user or user of a system. However A software requirements specification (SRS) is an elaborated description of a system to be developed with its functional and non-functional requirements along with the technical requirements. It consists of the use cases of how user is going to interact with system going to be implemented.

II. Implementation Detail

The implemented system has been divided into three core parts. First being the user story or software requirement specification part, second one is the NLP engine which is acted upon the first part to generate the third part that's nothing but the actual test cases. Two other parts included the test case writer utility and exporter utility which will export the test cases to portable formats. The figure below represents the workflow information about the implemented system.

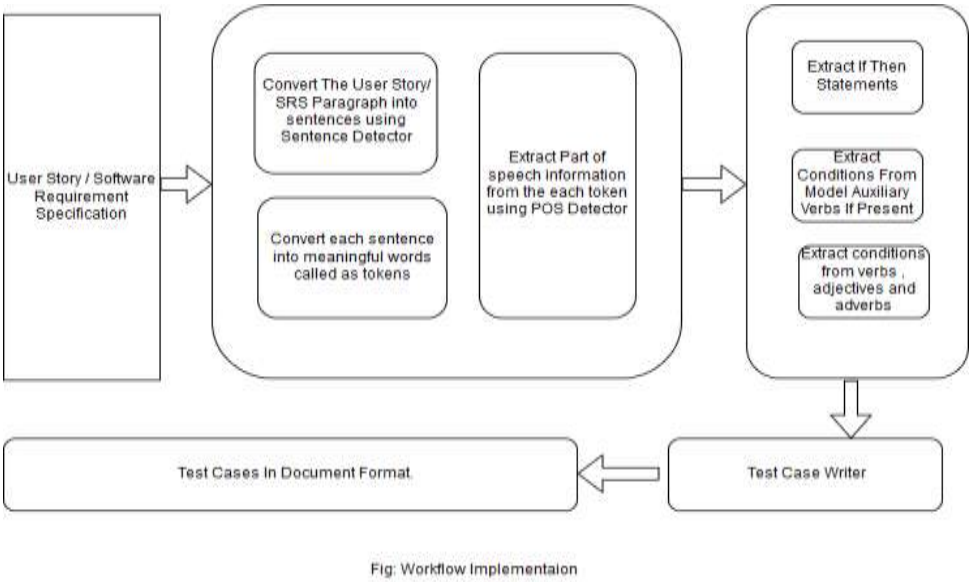

Fig. 1: Workflow Implementation Detail.

This implementation workflow works as follows. A user story or the Software requirement specification is feed to our system, one thing to clear here is that there is no particular format defined for it so a user story or SRS will be in human readable format.

After that the algorithm works as below:

i) From the user story/SRS, acceptance criteria and more like functional requirement if needed to cover the accetpaqnce criteria for that user story is extracted from it.
ii) The next step is to divide the acceptance criteria. It might be a paragraph or the group of the sentences. So each sentence need to be identified from the sentence to consider as single acceptance criteria item.
iii) To do this sentence detector model have been used here which is dividing the whole paragraph of acceptance criteria and functional or technical details into the list of statements.
iv) Going further this sentences need to be broken down into meaningful tokens which will further give more information
v) Hence sentence has been divided into the list of individual tokens.
vi) Along with tokens are also identified with their identities. Here it means part of the speech information.

vii) This also covers the recognition of the named entity which can be used as test data while testing the test case against the user story.
viii) This has been done using Part of speech detector model , in short called as POS detector model.
ix) The next step is to identify the condition from the individual statements.
x) To do this from the list if tokens conditional words like "If", "When" and "Then" has been checked if its present in the list of tokens if present sentence has been divided into the two parts and test case is written against it to validate the then condition against if condition.
xi) Sometimes instead of "Then", conditions are written by writing "If" and then expected result as separated by ",", so in this case a text after "," need to be validated against if condition.
xii) If there is no condition statement present, there might be chance that model auxiliary verbs might be used here. So here Part of speech information has been used to recognize the same.
xiii) From Part of speech information auxiliary verbs have been recognized, here again sentence can be divided into two parts one is before that auxiliary verb and other being after it. And test case can be constructed to validate the second condition against first.
e.g. "Published date of the blog should always be less than current date" this acceptance criteria point can be viewed as test case "Check if published date of blog is less than current date or not" where test data is blog and published date.
xiv) In rare case if no conditions and no model auxiliaries are used, in that case from the part of speech information of the sentence , information about the verbs have been found out and the same process has been carried out that has been done for the model auxiliaries.
xv) For the last two cases adverbs and adjectives have been also considered to get the more details.
xvi) Based on the all these information extracted, a test case writer utility will write the test cases on it ,and then will be converted into document format for further use.

The Figure 2 represents the overall block diagram of the implemented system. The individual model includes SRS/ User Story Feeder, Bug Tracking Tool, Core Natural Language Processing Engine and Finally Test cases.

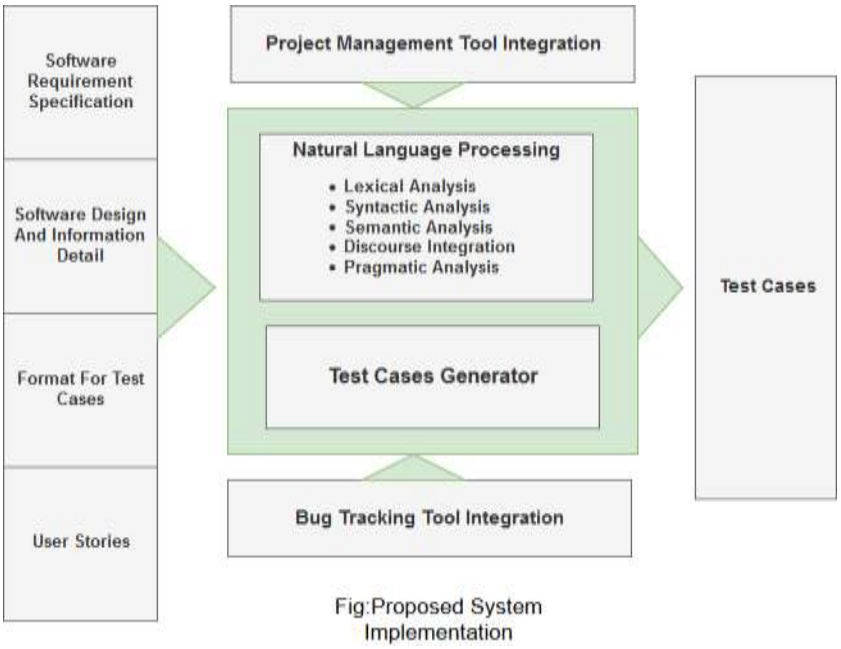

Fig 2: Implemented System Block Diagram.

Figure captions appear below the figure, are flush left, and are in lower case letters. When referring to a figure in the body of the text, the abbreviation "Fig." is used. Figures should be numbered in the order they appear in the text.

Consider a user story about the dynamic web application where one of the components lists the blogs in that application and it's having acceptance criteria as:

- Published date of blog should always be less than current date.
- If blog is associated with the property "archived" then it should not be listed.
- Clicking on the blog link will take you to appropriate blog page.
- Heading of the blog item should be the page title of the blog page.
- When the blog item is shown with the image, its featured blog.

So here, if this item passes through the implemented approach, the output generated is shown in the table given below.

Table 1: Test Case Generated By Algorithm and Performed By QA.

Test Step	Test Data	Expected Result	Actual Result	Status(Fail /Pass)
Check if published date of blog is less than current date or not	Blog, published date.	Published date of blog is less than current date.	Published date of blog is less than current date.	Passed
Check if blog is associated with property archived is not listed.	Blog, archive property	Blog with the archive property should not be listed.	Blog with the archive property is getting listed	Failed
Check if clicking on the blog is taking to appropriate page or not.	Blog, link	Clicking on the blog is taking to appropriate blog page.	Clicking on the blog is taking to appropriate blog page.	Passed
Check if heading of the blog item is page title of the blog page or not	Heading, blog, page title	Heading of the blog item is page title.	Heading of the blog is not page title.	Failed.
Check if the blog shown with the image is featured blog or not.	Blog, Featured blog, image.	Blog shown with the image is featured blog.	Blog shown with the image is featured blog	Passed.

Table 1 shows the test case generated using implemented system, the columns Actual result and status is filled by QA. Other columns items are generated using the implemented system. On Bug tracking tool if quality analyst will update this ticket, developer will come to know about the overall status and he/she will need to start the work on the steps which marked failed by QA.

III. Conclusion

Current Implementation shows the good use of the field of the Natural Language Processing to extract the useful information from the User story or Software Requirement Specification to construct the test cases automatically. This has reduced the chance of missing any important condition while writing the test cases manually. It further reduces the time spent on manual writing of the test cases. Implementation truly supports agile software development methodology which is the most used and trending software development model in current era.

It's integrated with the project management and bug tracking tool and hence project members can have direct tracking on the implemented system which is quite user friendly. Along with Quality analyst developers can also use this while performing the unit testing on their code. Test cases generated can be exported to different formats which can be used when working in offline mode.

References

[1] Ahlam Ansari ; Mirza Baig Shagufta ; Ansari Sadaf Fatima ; Shaikh Tehreem , Constructing Test cases using Natural Language Processing ,*Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB) 18(2),* 2017, 112-116.

[2] W.T. Tsai ; D. Volovik ; T.F. Keefe, Automated test case generation for programs specified by relational algebra queries, *IEEE Computer Society's International Computer Software & Applications Conference*, 1988

[3] 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),* 1997.

[4] 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.

[5] W.J. Book, Modelling design and control of flexible manipulator arms: A tutorial review, *Proc. 29th IEEE Conf. on Decision and Control*, San Francisco, CA, 1990, 500-506 (8)