# Comment Semantics Assessment Using Feature Location Techniques

Shilin N K
*Dept. of Computer Science*
Munster Technological University
Cork
shilin.n-k@mycit.ie

***Abstract*—In computer science and software engineering in particular source code is the core which decides the success of a software system. Understanding and analyzing a larger software system becomes challenging if adequate quality is not built in the code. Comments plays a major role here, it is a main source for documentation and helps to understand the system easily than knowing the code which is often complex. Comments contains vital information about a software system but developers often neglects commenting the code and fails to keep it up to date or miss its quality. It is important to have meaningful comments which semantically matches the source code to understand the software system. Although there are approaches to evaluate code comments and its quality exists, they ignore the comment semantics and its relation aspect with the code. This research work is going to fill that gap by assessing comment semantics. This will assess how relative is the comment with the code and does the comment accurately reflect the code. NLP based feature location techniques 'Latent Semantic Indexing' and 'Vector Space Model' are going to be applied on code comments to locate the feature. Successful navigation indicates project has good quality comments in place that semantically matches the source code. Low accuracy shows project requires improvements in the comments. This approach is going to extend further to assess whether the code or the comments has more accuracy in locating the feature.**

***Index Terms*—source code, comments, feature location, comment semantics, Latent Semantics Indexing, Vector Space Model**

## I. Introduction

Software systems meant to run longer so maintenance is an essential part of it. Its quality, security and design are often reviewed, analysed and refined through out the product life span. For software maintenance and to ensure code quality it is essential to address code plagiarism, identify feature similarity, find vulnerabilities, predict test failures etc and address those. Feature Location plays an important role in software maintenance including bug locating and fixing. But understanding and analyzing a large software system is highly challenging and if the system is legacy it becomes more complex, time consuming and budgetary. There have been various researches happening in this field to ease this process. Researchers keep exploring new methods and tools hence continuously contributing to the software engineering community.

Understanding the source code and its features are crucial in maintaining the software. Code is usually written once by a person but read multiple times by many. Comments allow developers to understand the code much faster and easier rather than going through the code which if often complex. Usually a significant part of the code contains comments written and evolved over the time. Researches shows that code with comment is easy to understand if it is written carefully without losing the code semantics [1]. Whether it is inspected manually or with programs comments are valuable entity that plays the role in software maintenance. There are many literatures work available with comments to locate features, to detect vulnerabilities, to assess its quality and to detect code clone etc. Its usability is endless, so it is essential to have good quality comments in place. At times it is not the quantity or the number of comments vs code ratio matters, rather the quality of comments, how semantically close these comments with the code matters the most.

This research explores a methodology to assess the comment semantics. This will judge how semantically close these comments with the source code and how accurately it can identify feature location. Result of this analysis will guide if a project requires any improvements in comments. Feature location technique is going to be used in the assessment. NLP based feature location technique Latent Semantic Indexing (NLP) and Vector Space Model (VSM) are going to be used for the analysis.

## II. Literature Review and Novelty

There are many literatures available in source code analysis however this proposal evaluates the most relevant and latest state of the arts in this field. Many literatures on source code comments exists today which addresses how to use comments to predict vulnerability, find technical debt, ensure coding standards, Feature location, assess code and comment quality and detect code clone etc. Researchers keep inventing new techniques and solutions to refine these areas that brings new novelty into this domain. They have been exploring potential of comments to automate various code analysis challenges to fasten the development curve. In recent years many NLP techniques have also been applied on comments to address those. Quantitative analysis of comments are usually expressed in the code comment ratio but that does not express its quality aspects. Semantically how close the comments are with the source code, this novelty is going to be addressed in this research.

Daniela Steidl and team has done their research on comment quality [1]. In that they assesses the quality aspect of comments based on various comment categories. Comments are classified as copyright comments, header comments, member comments, inline comments etc and measures its coherence with the code. Coherence refers its relation with the code, e.g. member comment should match with the method name. Coherence is the basis of quality in this model. Quantitative analysis of the systems comment ratio is also tailored with qualitative analysis to define the quality matrix. They have also conducted survey among 15 experienced developers to assess the quality matrix. Though this approach measures the quality of the comments it does not evaluate its semantic equality with the code. This novelty is the goal of this research. This research extends the quality aspect of comments further to assess its semantics by applying feature location techniques.

Comments are part of the program, but how do they evolve over the multiple versions of the software? Whether code comment ratio improves over versioning. All these questions are answered in the research 'Do Code and Comments Co-Evolve' done by Beat Flury and team [2].In this paper they map code and comments over multiple versions to plot its evolution. They have found newly added codes are less commented while class and methods are commented quite often. That research does the quantitative analysis of comments but its quality aspect is missing and that will be addressed by this research.

Code comments are processed using NLP to automatically detect technical debt [3] .Comments are tokenized, analysed using NLP then categorized based on design debt,development debts or requirement debts etc. This brings an automated approach of technical debt detection. This literature also proves comments contains vital information about a software product and can further develop to address other software analysis challenges. They were able to achieve 90 percent of the best classification performance using as little as 23 percent of the comments from the system, shows the capability of comments. This is further going to be developed in this research by doing feature location using comments.

Code clones are bad programming techniques and costs software maintenance. There are literatures proposed to detect code clone [4]. Usability of code comments are exposed through this, with the help on NLP comments are analysed to locate the code clones in the system. Another research has been carried out by a team of researchers that does deep evaluation and assessment of code comments in a software system [5].Code and comments are passed to a multi-input neural network and processed by a Perceptron classifier to assess the comment quality.

Code context is used in predicting comment location [6]. These literatures undoubtedly proves the association with code and comments, and they can be used together to solve software engineering challenges. Similarity of Source Code has been analysed in another paper in the Presence of Pervasive Modifications [7]. Code comments further explored to boost program Comprehension [8] and helps programmers understand a nontrivial part of source code. All these shows the power of comments in a software system. This research is going to assess the novel aspect of its semantics with code, also going to assess whether code or comments is more accurate in locating the feature and recommend if a project needs improvements in its comments.

## III. Research Aim

Semantic assessment of comments by applying NLP based feature location techniques Latent Semantic Indexing and Vector Space Model.

## IV. Research Objectives

### *A. Assess whether the artifacts or the comments have more accuracy in locating the feature*

Artifacts or artifacts with comments are widely used in feature location techniques. Novelty of this research is to assess whether comments or artifacts have more accuracy in feature location when applied NLP based feature location techniques. [1]

### *B. Assess the quality of comments to semantically match the source code*

Most times it is the quality of comments - that is semantically how close the comments with the code matters than its quantity. It helps developers in trouble shooting and understand the system faster. Good comments gives more accurate feature location results.This research is going to assess the semantic aspect of comments with the code for feature location.

### *C. Assess whether removing code-commented improves the quality of the Feature Location*

It is very common to have commented code in a software program. Often developers keep the commented code for future reference, temporarily commented or they may even forget to remove it altogether. This research is going to assess how the commented code impacts the feature location results. Does it going to improve the accuracy or will that impact the result negatively.

### *D. Determine if a project requires improvements in comments*

A software program is a mixture of code and comments. Comments may appear anywhere in the code, line comment, document comment, method comments etc. When there are lots of comments in the code visual inspection might give a feel of adequate comments but that may not be the truth. A programmatic assessment gives more accurate result. By applying feature location techniques on comments to find its accuracy is going to assess whether comments needs improvement or not.

### *E. Explore Feature location techniques LSI or VSM has more accuracy when working with comments*

LSI is a variant of VSM and both are widely used technologies in Information Retrieval and one of its application is feature location. This can be applied on comments, artifacts or together to locate the feature. This research is going to explore which one results more accurate feature location when working with comments.

*During the detailed scrutiny of the research content and the challenge involved in developing the software systems to assess these objectives it was evident that all these research questions can be answered only if adequate time is available. Limited semester timelines and any unforeseen external events may have an impact on the research scope.*

## V. Research Methodology

The proposed research will be following Qualitative methodology to fulfill the research objectives. Primary goal of this research is to assess how semantically close the comments are with the source code and its usability in feature location. NLP based feature location technique LSI and VSM are going to be used in this research. Comments and artefacts are processed individually and together as well to find accurate feature location for the query. This methodology is refined further to find answers for the research questions mentioned above. If comments alone is able to locate the feature in more accurate that shows project is having a quality comments in place and it is semantically matches the source code.

Various programming exercises needs to be conducted in each stage of the analysis to reach the required outcome. Developed program needs to be tested and validated against the gold set. A GitHub project has been identified as a gold set to validate the program. (https://github.com/masud-technope/BLIZZARD-Replication-Package-ESEC-FSE2018/tree/master) [9]. Once the program is validated and tested against the gold set it will then be applied on various open source projects.

Following are various implementation phases. Each phase is implemented as an independent project similar to micro services and each phase's output is key-in as input to the next phase. This brings modular design and loose coupling of components that enables independent development, testing and integration of different phases of developments.

### *A. Phase 1: Java Parser - Parsing java files and creating documents*

JavaParser library is used to parse the java source files. Each file is parsed to extract the comments - line comment, method comments, document comments all identified in this step. This data goes to NLP filtering such as tokenize, stop word removal etc to create a document with all these fields. Various documents are created in this phase such as document with comments, document with artifacts, document after removing code-commented. These different documents are used in the phase2 and phase 4 for processing and validation of feature location.

### *B. Phase 2: Processing - Applying Feature Location Technique LSI and VSM*

Output of JavaParser module is the input to this phase. Documents created by Phase1 are processed using LSI and VSM Feature location techniques to find the most matching document. This processing is further refined for various combination of documents such as document with comments, documents with comments and artifacts, document after removing code-commented to generate data for the research questions assessment.

### *C. Phase 3: Validation - Validating the program against gold set*

Previous phase gives the feature location result. This is validated against the gold set to ensure the program functions as expected. Result from previous steps based on each document is validated in this phase with gold set to qualify the program for the next phase.

### *D. Phase 4: Evaluation - Evaluation of comment semantics and research questions*

This is the final phase where comment semantics and various research questions are answered. Qualified program from the previous phase is applied on identified open source projects in addition to the gold set to assess the code comments and feature location. This is the phase where research questions are answered.

Below diagram shows the basic flow of research methodology.

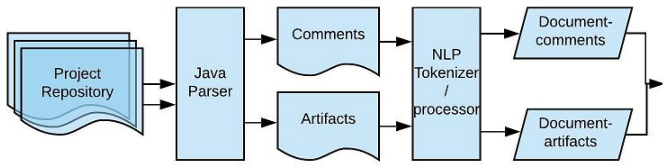


Fig. 1. Methodology Fig 1

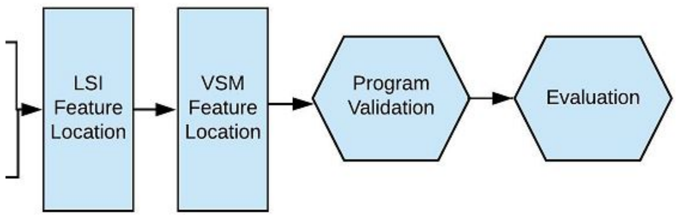


Fig. 2. Methodology Fig 1 - continuation

## VI. Timelines and Work Plan

Research project spans over four months, starting from January 2021 till mid May. A research execution plan is set with the help of WBS and Gantt chart to meet the time lines. Project execution progresses through various phases as explained below.

Research will be conducted in a series of phases. Learning and data gathering phase is the first one where deep analysis of latest literatures and technology are carried out. Various literatures in the field of feature location and code comments are reviewed to learn more about this researched items. Finding the right gold set to validate the program is the next task. Gold set needs to be finalized and its readiness is important for the project validation. Next phase is the development phase where the program is developed to find the feature location for the given search string. After the initial validation same will be tested against the gold set to assess the program quality. Once verified a few open source project will be assessed using this program to know whether it has meaningful comments in place or it requires improvements. This will be iterated on various projects to assess its quality.

In this development phase first part is the code parser. JavaParser library will be used to parse the java source files to create documents from it. This follows processing phase to find feature location and then program validation phase. After the initial validation of the program it will then be tested against the gold set to assess its readiness and quality.

Result evaluation and the thesis submission are the final two phases of the project. Results from the previous phase is evaluated here by applying the program over identified open source projects. Results are collected, reviewed and prepare the matrices to conclude the research questions. Final dissertation document begins in parallel to this. Documentation is an ongoing tasks through out these phases and final report then reviewed and submitted at the end of the thesis.

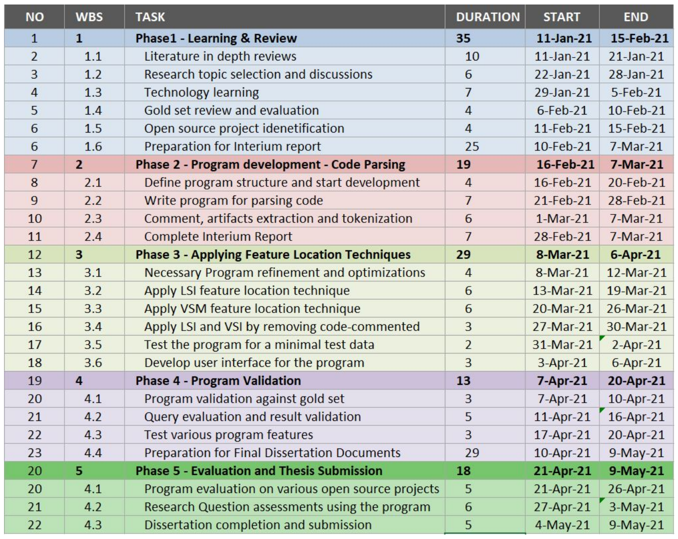
Below detailed the work breakdown structure (WBS) of the research project.

A Gantt chart has been prepared based on this WBS. This chart is the visual representation of the WBS breakdown that shows when a task is starting, when it finishes and also shows the overlapping tasks such as documentation.

## VII. Current Progress of this research

### A. *literature reviews completed*

Following litterateurs have been reviewed to know the latest state of the art in the field of comments processing and feature location. Research questions are formulated based on the learning from these papers. It is then discussed with professor to refine and tune the research topic. These are some of the literature reviewed as part of this project. Quality analysis of source code comments [1], Do Code and Comments Co-Evolve? On the Relation between Source Code and Comment Changes [2], Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt [3], Semantic Clone Detection: Can Source Code Comments Help? [4], Deep Code-Comment Understanding and Assessment [5], Learning Code Context Information to Predict Comment Locations [6], Similarity of Source Code in the Presence of Pervasive Modifications [7] and Analyzing Code Comments to Boost Program Comprehension [8]



| NO | WBS | TASK | DURATION | START | END |
|---|---|---|---|---|---|
| 1 | **1** | **Phase1 - Learning & Review** | **35** | **11-Jan-21** | **15-Feb-21** |
| 2 | 1.1 | Literature in depth reviews | 10 | 11-Jan-21 | 21-Jan-21 |
| 3 | 1.2 | Research topic selection and discussions | 6 | 22-Jan-21 | 28-Jan-21 |
| 4 | 1.3 | Technology learning | 7 | 29-Jan-21 | 5-Feb-21 |
| 5 | 1.4 | Gold set review and evaluation | 4 | 6-Feb-21 | 10-Feb-21 |
| 6 | 1.5 | Open source project idenetification | 4 | 11-Feb-21 | 15-Feb-21 |
| 6 | 1.6 | Preparation for Interium report | 25 | 10-Feb-21 | 7-Mar-21 |
| 7 | **2** | **Phase 2 - Program development - Code Parsing** | **19** | **16-Feb-21** | **7-Mar-21** |
| 8 | 2.1 | Define program structure and start development | 4 | 16-Feb-21 | 20-Feb-21 |
| 9 | 2.2 | Write program for parsing code | 7 | 21-Feb-21 | 28-Feb-21 |
| 10 | 2.3 | Comment, artifacts extraction and tokenization | 6 | 1-Mar-21 | 7-Mar-21 |
| 11 | 2.4 | Complete Interium Report | 7 | 28-Feb-21 | 7-Mar-21 |
| 12 | **3** | **Phase 3 - Applying Feature Location Techniques** | **29** | **8-Mar-21** | **6-Apr-21** |
| 13 | 3.1 | Necessary Program refinement and optimizations | 4 | 8-Mar-21 | 12-Mar-21 |
| 14 | 3.2 | Apply LSI feature location technique | 6 | 13-Mar-21 | 19-Mar-21 |
| 15 | 3.3 | Apply VSM feature location technique | 6 | 20-Mar-21 | 26-Mar-21 |
| 16 | 3.4 | Apply LSI and VSI by removing code-commented | 3 | 27-Mar-21 | 30-Mar-21 |
| 17 | 3.5 | Test the program for a minimal test data | 2 | 31-Mar-21 | 2-Apr-21 |
| 18 | 3.6 | Develop user interface for the program | 3 | 3-Apr-21 | 6-Apr-21 |
| 19 | **4** | **Phase 4 - Program Validation** | **13** | **7-Apr-21** | **20-Apr-21** |
| 20 | 4.1 | Program validation against gold set | 3 | 7-Apr-21 | 10-Apr-21 |
| 21 | 4.2 | Query evaluation and result validation | 5 | 11-Apr-21 | 16-Apr-21 |
| 22 | 4.3 | Test various program features | 3 | 17-Apr-21 | 20-Apr-21 |
| 23 | 4.4 | Preparation for Final Dissertation Documents | 29 | 10-Apr-21 | 9-May-21 |
| 20 | **5** | **Phase 5 - Evaluation and Thesis Submission** | **18** | **21-Apr-21** | **9-May-21** |
| 20 | 4.1 | Program evaluation on various open source projects | 5 | 21-Apr-21 | 26-Apr-21 |
| 21 | 4.2 | Research Question assessments using the program | 6 | 27-Apr-21 | 3-May-21 |
| 22 | 4.3 | Dissertation completion and submission | 5 | 4-May-21 | 9-May-21 |

Fig. 3. WBS Figure

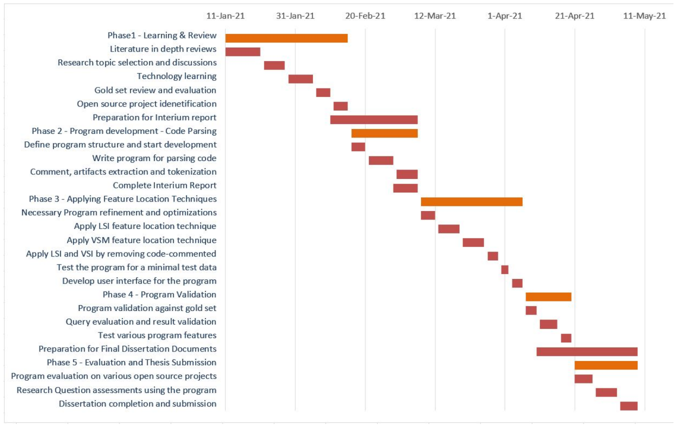


Fig. 4. Gantt Chart

### B. *Gold set Identified*

Identified the gold set https://github.com/masud-technope/ BLIZZARD-Replication-Package-ESEC-FSE2018/tree/master/ [9] This gold set is extracted from the github project 'BLIZZARD-Replication-Package-ESEC-FSE2018' - A research paper 'Improving IR-Based Bug Localization with Context-Aware Query Reformulation' by Masud Rahman. In this paper he has introduced a new technique BLIZZARD– that auto localizes bugs from source using query reformulation and information retrieval. This projects uses query based feature location to locate the defect. This is going to be treated as a gold set for this research and the program will be validated against this.

Baseline/Query/ecf.txt contains various queries. Each query is labeled with a number e.g 112599, 119206. These numbers are important as the query result is going to have the same number. e.g query number 112599 is ”XMPP Room subject updated xmpp chat updated remotely xmpp server title room updated dynamically” and its result is going to be in 11259.txt file. Corpus/ecf contains java files. Original file names are renamed to numbers. e.g. 100.java, 1001.java etc. Its number to original filename mapping is available in ecf.ckeys index file. Goldset/ecf is having the query result and it is mapped according to the query number. e.g 112599.txt file is the query result for the query id 112599. . Its result is XMPPChatRoomContainer.java which is mapped in ecf.keys with the file name 2220.java

### *C. Open source projects have been identified for the evaluation*

Popular open source projects written in java are identified for the theses evaluation in addition to the gold set. Research questions are answered here by running the software programs against these projects and evaluating the results.

Antlr4(https://github.com/antlr/antlr4/tree/master/)
Selenium(https://github.com/SeleniumHQ/selenium/)
Apache-Groovy( https://github.com/apache/groovy/tree/master)
Kafka( https://github.com/spring-projects/spring-kafka/)
Cloudfoundry Uaa( https://github.com/cloudfoundry/uaa/tree/)

### *D. Parsing of source code*

Initial phase of the program is parsing the code with JavaParser and this has been completed. A program is written to parse the java files using JavaParser library. Files are parsed using the library then extracted the comments using *getAllContainedComments()* API of the JavaParser library. These comments go through NLP pre-processing such as tokenize, stop word removal etc then added as a ’comment’ field into the document. These documents are then passed over to the next module for applying feature location.

## VIII. Ethical Issues

Following all the primary ethics but not limited to have been considered in the research;

- All reference materials and quotes are cited appropriately

- Respect and dignity has given for each researcher in the filed of source code analysis

- Adequate level of confidentiality of the research is maintained

- There are no affiliation, funding or grands provided for this research

- This research will be conducted for the academic purpose associated with MTU only

- There are no misleading or biased information present in the research

- Research is adhered to the data protection rule GDPR.

## References

[1] D. Steidl, B. Hummel and E. Juergens, ”Quality analysis of source code comments,” 2013 21st International Conference on Program Comprehension (ICPC), San Francisco, CA, USA, 2013, pp. 83-92, doi: 10.1109/ICPC.2013.6613836.

[2] B. Fluri, M. Wursch and H. C. Gall, ”Do Code and Comments Co-Evolve? On the Relation between Source Code and Comment Changes,” 14th Working Conference on Reverse Engineering (WCRE 2007), Vancouver, BC, Canada, 2007, pp. 70-79, doi: 10.1109/WCRE.2007.21.

[3] E. d. S. Maldonado, E. Shihab and N. Tsantalis, ”Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt,” in IEEE Transactions on Software Engineering, vol. 43, no. 11, pp. 1044-1062, 1 Nov. 2017, doi: 10.1109/TSE.2017.2654244.

[4] A. Ghosh and S. K. Kuttal, ”Semantic Clone Detection: Can Source Code Comments Help?,” 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Lisbon, 2018, pp. 315-317, doi: 10.1109/VLHCC.2018.8506550.

[5] D. Wang, Y. Guo, W. Dong, Z. Wang, H. Liu and S. Li, ”Deep Code-Comment Understanding and Assessment,” in IEEE Access, vol. 7, pp. 174200-174209, 2019, doi: 10.1109/ACCESS.2019.2957424.

[6] Y. Huang, X. Hu, N. Jia, X. Chen, Y. Xiong and Z. Zheng, ”Learning Code Context Information to Predict Comment Locations,” in IEEE Transactions on Reliability, vol. 69, no. 1, pp. 88-105, March 2020, doi: 10.1109/TR.2019.2931725.

[7] C. Ragkhitwetsagul, J. Krinke and D. Clark, ”Similarity of Source Code in the Presence of Pervasive Modifications,” 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM), Raleigh, NC, USA, 2016, pp. 117-126, doi: 10.1109/SCAM.2016.13.

[8] Y. Shinyama, Y. Arahori and K. Gondow, ”Analyzing Code Comments to Boost Program Comprehension,” 2018 25th Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan, 2018, pp. 325-334, doi: 10.1109/APSEC.2018.00047.

[9] @INPROCEEDINGSfse2018masud, author=Rahman, M. M. and Roy, C. K., booktitle=Proc. ESEC/FSE, title=Improving IR-Based Bug Localization with Context-Aware Query Reformulation, year=2018, pages=621-632