# Report on A Concept Location Tool (A research paper implementation)

Course SE 801: Project

Submitted By:

Dipok Chandra Das
BSSE0501
IIT, DU

Supervised By:

Mr. Rayhanur Rahman
Lecturer
IIT, DU

Submitted to:

BSSE 4th year Program Committee
IIT, DU

Date of Submission November 15, 2016

## *Abstract*

Concept Location is a part of incremental change. The developers frequently undertake the concept location process for software maintenance and evolution. When a change request or bug report comes, the developers first find out the place in source code where they should work on to mitigate the change request or bug report. They should have enough knowledge about the source code for finding the place in source code. But it becomes difficult for a large system to know where the concept related to that change or bug request being implemented in source. It can be seen as they search for specific information in source code. Marcus et al [8] proposed Information Retrieval (IR) approach for concept location. In this approach, the source code is treated as collection of documents. The developers search for information with a query representing a concept. IR returns a set of documents from collection of documents best matched with the query. IR-based concept location has several advantages over traditional ***grep*** based technique and dependence graph. IR approach saves time. It also helps the developers to know about an unknown software system. Several extensions are proposed in IR-based approach including relevance feedback, clustering, and formal concept analysis. So, a concept location tool is developed using the IR approach and IR with Relevance Feedback approach to solve the search problem for concept location. Relevance Feedback produces better result than IR approach.

## Table of Contents

Table of Figures

## 1 Introduction

In Software maintenance and evolution, Concept Location process maps concepts into software components (Fig 1). A concept is extracted by the developers from a maintenance request. The input to concept location process is a concept expressed in natural level language and the output is the set of software components that implement the concept.

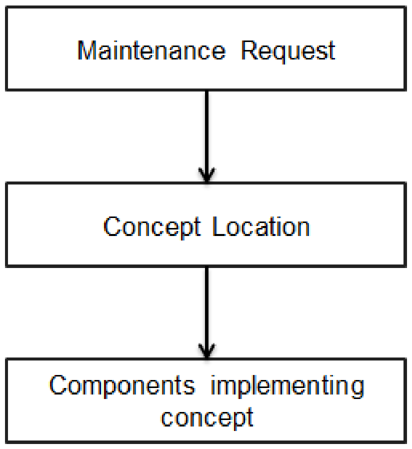

Fig 1 : Concept Location Process

A software change request is actually a new functionality or a modification of existing functionality of a system. If it is a modification of existing functionality, the developers find out the place where it was implemented. If it is a new functionality, the developers think to adapt the new functionality. They first find out the place which parts of code will accept the new functionality. A bug report is like a functionality which is implemented but not working as expected or needs to modify. So, to solve this bug, a developer has to find out the function in source code. To find out the parts of source code that implements a specific concept is called concept location. External Documentation actually does that mapping between concepts and source code. In absence of external documentation, it becomes a search activity for the developers. Concept Location can be done by static or dynamic technique. In static technique, the static behavior of source code is used; the execution of source code is not needed. The dynamic technique uses the dynamic behavior actually the execution of system for concept location. Static techniques use the lexical or structural information of the source code or both.

The developers from their intuition search source code to find out those parts representing concepts. It is easy when the size of a software system is low. But when the size increases it becomes a difficult task because search space of systems increases. The developers generally use the string pattern matching approach in finding a variable or a term that can be extracted from a concept. *grep* (global regular expression pattern) is widely used in string pattern matching approach. If a pattern matching occurs, the developers see that surrounding area to take decision whether the concept is located or not. In this approach, the developers have to examine lots of matching with same priority. Another concept location process is dependency graph. This approach uses the structural information of source code. To find out a specific concept, the developers have to go through the full dependency graph. It is a depth search problem.

If a newly joined developers is given the task of fixing a bug who knows the functionally of software but unknown to the source code, it becomes difficult for him. He does not know about the identifiers. He manually checks each class, each method. It is a time consuming task. It can be thought of that each method or class has some information. A concept is a topic which is elaborately written in classes or methods. When the information gained from source represents the concept, it is called a concept is located. Marcus et al [8] addressed concept location process as an information retrieval problem. Here the source code is a collection of documents. For a specific information need, the developers will search the collection of documents with a query. The information retrieval method will return a ranked document list based on the query. The developers will inspect the ranked documents whether a document meets the concept they are looking for or not. Here, the developers can search for a concept in source code in a ranked ways so the search space becomes narrow than previous steps like string pattern matching or dependency graph. If a developer does not know about anything, he can gain knowledge from the returned results. The better the query, the better the result is. A good query is generated from good knowledge about the domain. If a newly come developer does not know about the system there exists an information retrieval method called Latent Semantic Indexing to generate a query automatically. There are several IR methods. The two common methods are Vector Space Model and Latent Semantic Indexing.

In IR approach, the developers get search result based on query. From the returned results, the developers gain knowledge about the system. They update the query by adding some terms or removing terms. They again search the source with updated query. This improves the search results. But if the developers does not know about the terms used in source code to represent concept, relevance feedback helps the developers to get the desired the parts of source code. Marcus [4] proposed IR with Relevance Feedback for concept location. In this approach, the query is updated automatically on relevance feedback. The updated query generates a new search result. Rocchio Algorithm is widely used for relevance feedback.

## 2 Background

### 2.1 Information Retrieval

Information retrieval (IR) is retrieving information from a collection of information. It is actually retrieving information from a huge collection of information based on user query. Information retrieval can textual or non-textual. In Information Retrieval, the collection of information is pre-processed and indexed by different methods. The user query is processed and translated by IR so that best relevant information can be retrieved.

### 2.2 Vector Space Model

Vector Space Model [16] is popular in text retrieval Information Retrieval method. Vector Space Model relies on the bag-of-words model.

Each document in the collection is converted into a vector called document vector. Each document vector has m components where m is the total number of terms present in the bag-of-words. The score of a term in a document vector is calculated from term-frequency and inverse-document-frequency. Term-frequency (tf) is the frequency of a term in the document. Inverse-document-frequency (idf) is calculated by document frequency (df) and the number of documents in the collection. Document frequency for a term is the number of documents where the term occurs. There are some variations in calculating idf. One of them is,

$$idf(t,d) = 1 + \log(df/n) \qquad \text{………… Equation 2.1}$$

, where n is the number of total documents

The low value of df and resulting higher value of idf indicates that the term in that document is significant.

Now,

$$tf - idf = tf * (1 + \log(df/n)) \qquad \text{………… Equation 2.2}$$

The similarity between two documents is calculated by the dot product or by cosine measure of their corresponding documents vectors.

$$dot - product = V(d_i) \,.\, V(d_j) \qquad \text{………… Equation 2.3}$$

Cosine measure is,

$$sim(d_i, d_j) = \frac{V(d_i) . V(d_j)}{|V(d_i)|.|V(d_j)|}$$ ............. Equation 2.4

Cosine measure is used for length normalization of two documents. Higher cosine measure indicates higher similarity between two documents.

When the query can be thought like a document. The cosine measure between query vector and document vector is calculated to find the similarity between query and a document. If query is expressed as vector, $V(q)$ then $sim(d_i, q)$ is calculated by the Equation 2.4.

Vector Space Model works on bag-of-words. It does not deal with the order of words. Here, “Saif is taller than Shaon” or “Shaon is taller than Saif” is same. The terms are independent.

## 2.3 Latent Semantic Indexing

Latent Semantic Indexing is another IR method. A VSM model suffers from synonym and polysemy problem. Latent Semantic Indexing [1] solves these problems. LSI does not take the terms are independent. The association of terms and documents are used to find the semantic structure.

## 2.4 Dependence Graph

In the C language, the System Dependency Graph (SDG) [2] consists of nodes that represent components, i.e. functions and global variables. Function call is represented by call edge and data flow edge represents flow of data from a function to a global variable and vice versa.

## 2.5 BorderFlow Algorithm

BorderFlow [3] is a general-purpose graph clustering algorithm. A BorderFlow algorithm generates clusters in a manner such that the nodes in one cluster have more links than links to the nodes of another cluster. It is a soft cluster. A node can present in several clusters.

## 2.6 Relevance Feedback

Information Retrieval method retrieves documents from collection on a user query. Relevance feedback on the retrieved documents by the user helps the system better understanding the user query. So, the performance of the system increases. Explicit, implicit, and blind (“pseudo”) feedback are three type of relevance feedback. Explicit feedback is taken from the user. The feedback can be in binary form (relevant or irrelevant) or graded system (4 out of 5, 2 out of 10, etc).

## 2.7 Rocchio Algorithm

Rocchio algorithm [16] is used in Relevance Feedback with Vector Space Model technique. In Relevance Feedback, the query is updated on user feedback. Query is updated in a way so that the similarity between query vector and relevant documents is maximized and the similarity between query vector and irrelevant documents is minimized.

If Q is a query vector and there are r number of relevant documents denoted by R and i number of irrelevant vectors denoted by I. And D is the total collection of documents then the new query,

$$Q' = \alpha * Q + (\frac{\beta}{r}) * \sum_{j=1}^{r} \mathrm{R} - (\frac{\gamma}{i}) \sum_{j=1}^{i} \mathrm{I} \qquad \text{....... Equation 2.5}$$

$\alpha, \beta, \gamma$ are weighting parameters.

## 2.8 Formal Concept Analysis

Formal concept analysis (FCA) [5] is to identify meaningful groupings of objects that share common attributes. FCA can be defined as C = (A, O, R) where O is set of all objects, A is a set of all attributes and R is a binary relation R ⊆ O x A. A formal context is a collection of concepts. Each concept is a maximal collection of objects that has common attributes.

# 3 Literature Review

To identify the parts of source which implement system functionality is a pre-requisite to program comprehension. It is most common activities undertaken by the developers [6]. It is referred as concept location. The process is also called concept assignment problem [7]. In software maintenance and evolution, a change is generated by change request. A change request can be a new feature or bug report [8]. In order to make the change, the developers first find out the parts of the code where a specific concept is implemented.

The input in a concept location process is a concept generated from problem domain and the output is the software components that implement the concept in solution domain. In ideal case, traceability link between documents and source code provide all that is needed for concept location. But in practice, in most cases, the documentation is outdated and does not exist. Concept location techniques use two type of information [3]. One is static and another is dynamic. Dynamic Information is extracted from the execution of the system. On the other hand, Static information is extracted from the source code without executing the source code.

Wilde [9] developed the Software Reconnaissance method which uses dynamic information to locate features in existing systems.

Rajlich and Wilde [10] [11] analyze the importance and role of concepts in program comprehension, describe the process of concept location, and discuss how vital it is to the maintenance of code.

Eisenbarth [12] uses dynamic information gathered from scenarios of invoking features in a system. The analysis of execution traces is geared towards feature location. Features are special concepts that describe the system functionality, observable at execution.

## 3.1 Concept Location Techniques

Concept Location is an informal and intuitive process frequently undertaken by the developers. It is based on previous knowledge about software. Usually the developers navigate the whole source program to find where a specific concept is implemented. When the software is small, it is an easy task. But it becomes a daunting task for complex large software. When the developers fail with intuition and experience, they go for string based pattern based approach. This approach takes the advantage the similarity of concept names and identifiers name. The developers read the code surrounding a matching to decide whether the code reflects the concept or not. The developers usually use ***grep*** based string pattern matching technique.

This technique has known weakness. When a concept is hidden in code or the developers use synonyms, this technique fails. Marcus and Rajlich [8] also stated that in this technique, the developers have to examine a large set of documents with same priority.

Chen and Rajlich [2] proposed an approach for locating concepts based on the search of program dependency graph. This technique used varied search expansion techniques including top-down or bottom-up and forward or backward data flow strategies. This technique reduces the search space for locating concept in source program. But the development of a dependence graph is difficult.

Marcus and Rajlich [8] an Information Retrieval (IR) based approach to Concept Location. In this approach, the source code is considered as a corpus. The corpus is indexed by Latent Semantic Indexing (a method of IR). They formulate query and search the systems.

## 3.2 IR Based Concept Location

IR based approach to Concept Location was proposed by Marcus and Rajlich [8]. The IR-based approach is composed of five major steps. The steps are as follow:

Step 1: Corpus creation

The collection of documents is extracted from the source code by parsing comments and identifiers. The granularity can be method or class level.. Each method or class corresponds to a document. The collection of documents is the corpus.

Step 2: Indexing

The corpus is indexed using the IR method. Each document (method or class) has a corresponding index.

Step 3: Query formulation

The developers can manually generate a query that represents a concept. There is also way of generating query automatically.

Step 4: Ranking documents

The similarity measure between the query and each document in the corpus is computed. Based on the similarity measure, the documents in the corpus are ranked.

Step 5: Results examination

The developer examines the ranked list of source code documents. The developer decides whether a document is in the ranked documents is relevant or not to his query (see Fig 2).

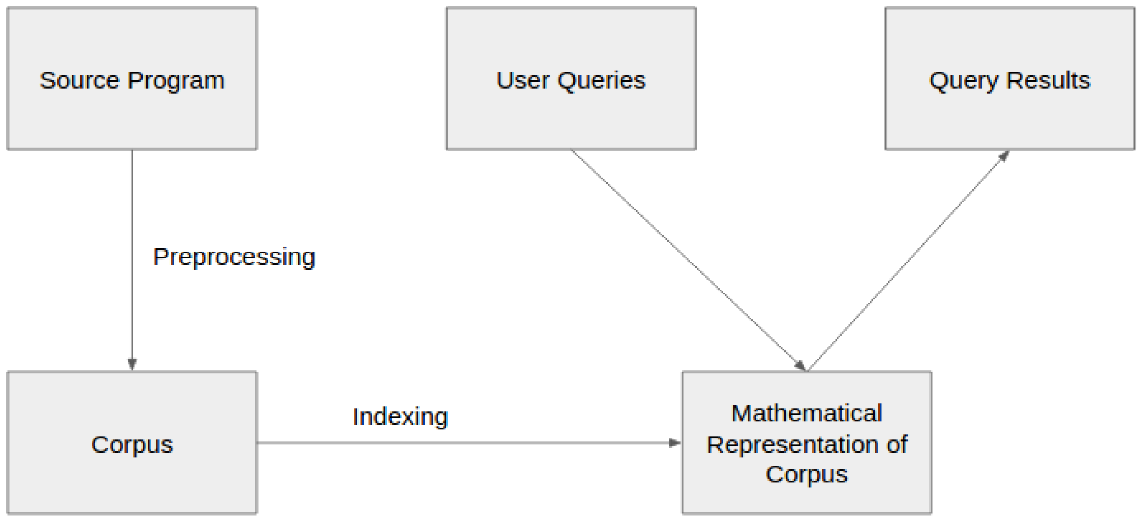

Fig 2: IR based Concept Location Process

Marcus and Rajlich [8] showed that IR based approach have several advantages over ***grep*** and dependence based approach. Based on the basic steps, the next work was done adding some additional steps.

## 3.3 Combining IR and Formal Concept Analysis for Concept Location

In IR based approach, the technique returns a relevant ranked list of documents. The developers examine the documents to find out the appropriate document implementing a specific concept. It is also a difficult task for the developers. Marcus [6] [5] combined Formal Concept Analysis (FCA) to reduce the search effort (Fig 3). In this approach, the technique selects the most relevant attributes from the ranked list of documents and generates a labeled concept lattice by FCA. The developers can determine a node of concept lattice is relevant or irrelevant simply by examining its label. The developers can also ignore the irrelevant nodes. It reduces the search space for the developers. So now the steps are as follow [5]:

Step 1: Corpus Creation – same as IR
Step 2: Corpus Indexing – same as IR
Step 3: Query Formulation – same as IR
Step 4: Query Results – same as IR
Step 5: Selecting descriptive attributes
The top k attributes from the first n documents in the ranked list are selected.
Step 6: Applying Formal Concept Analysis
Now the formal context is generated from n documents (objects) and k terms (attributes). Then the concept lattice is constructed from the formal context.
Step 7: Examining results
The developer now examines the nodes of a concept lattice. Each node in the concept lattice corresponds to a concept. The developer examines the documents.

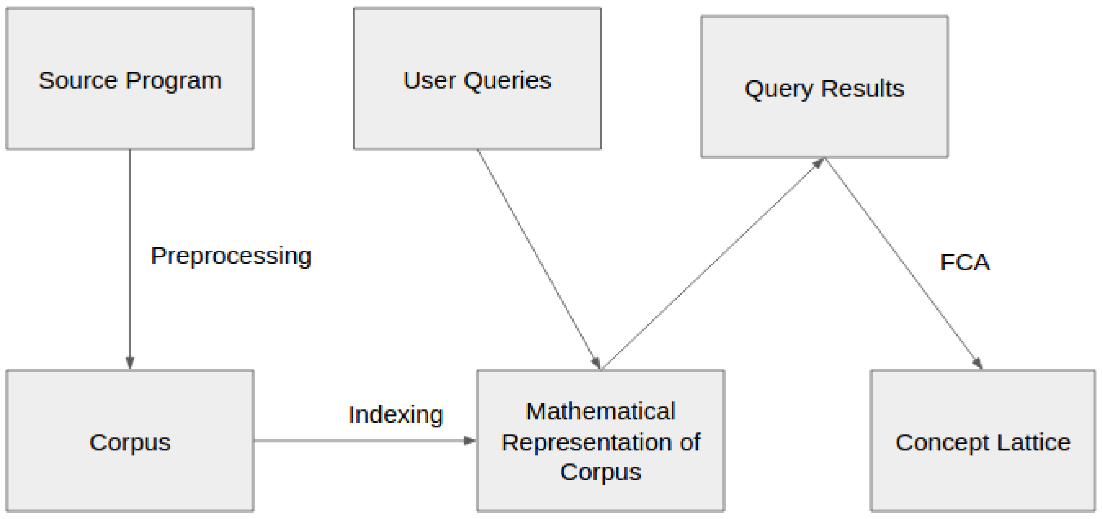

Fig 3: Combining Formal Concept Analysis with IR

## 3.4 Relevance Feedback in IR-based Concept Location

Marcus [4] proposed explicit Relevance Feedback in IR-based concept Location (Fig 4). In this approach, the developers judge the returned documents. The IR system uses this information to generate new search and return new search results. By repeating this judgement and search result, the developers look for his desired documents.

The new concept location methodology is defined as follows [4]:

1. Corpus creation – same as IR.
2. Indexing – same as IR.
3. Query formulation – same as IR.
4. Ranking documents – same as IR.
5. Results examination

The developer examines the top ranked documents. If the developer finds his desired document, then concept location process stops. But if he does

not find that wants to give feedback to update his query. Then Step 3 is executed. Several rounds of feedback are done to get the desired result. If unsuccessful, then query is generated manually by the developer and Step 4 is executed.
Marcus used Rocchio [4] relevance feedback method.

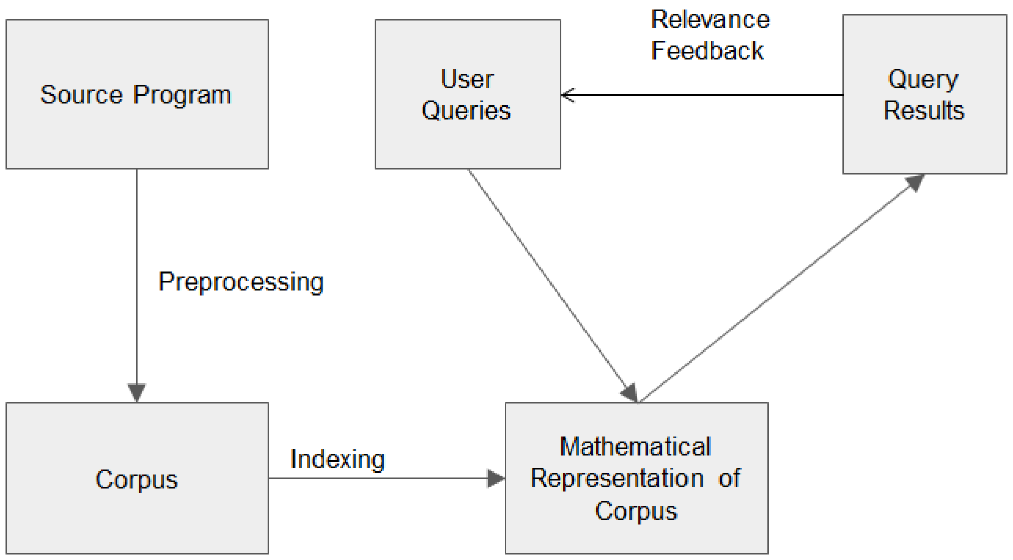

Fig 4 : Relevance Feedback in IR based concept location

## 3.5 Clustering Support for Static Technique of Concept Location

These are lexical (i.e. text present in the source code) and structural (i.e. dependencies present in the source code) information used for concept location. Marcus [3] proposed an approach which combines both lexical and structural techniques to perform textual search on source code. The approach clusters the source code. The developers get the clusters of documents to examine instead of ranked list of documents. The approach uses BorderFlow algorithm to prepare the clusters with both lexical and structural information. Now new concept location technique is as follows [3]:

Step 1: Corpus Creation – same as IR
Step 2: Indexing - same as IR
Step 3: Semantic similarities between source code documents.
The semantic similarities between source codes are computed by IR method.
Step 4: Dependencies
Dependencies are extracted from the source code using static method references.
Step 5: Clustering
The software system is clustered using the lexical similarities between source code documents and the structural dependencies between them. Marcus [3] used  BorderFlow clustering algorithm.
Step 6: Formulate a query – same as IR
Step 7: Ranking documents
Then lexical similarities between the query and the documents are computed. Similarities between the query and each cluster are also computed. Clusters are retrieved based on their similarity to the query. Inside each cluster, the documents (methods) are ranked by their similarity to the query in descending order.
Step 8: Examining results.
The developer examines the results. If a user finds a part of the concept then the search succeeds, otherwise, the user formulates a new query, returns to step 7.  See Figure 6.

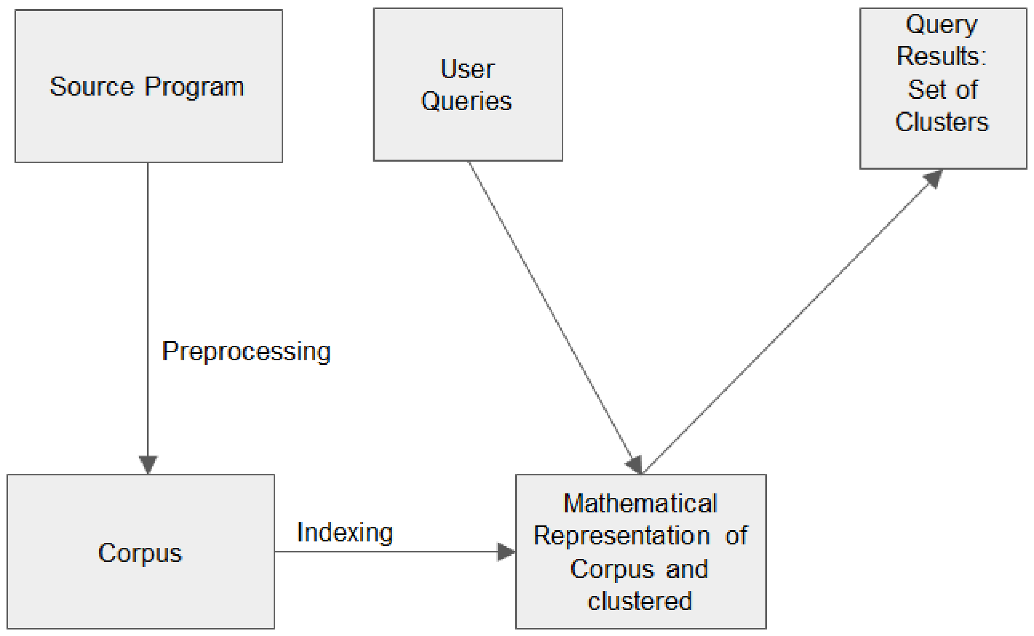
Source Program
User Queries
Query Results: Set of Clusters
Preprocessing
Corpus
Indexing
Mathematical Representation of Corpus and clustered

Fig 5 : Clustering support for Static Techniques in Concept Location

## 3.6 Query formulation in IR based Concept Location

A query is an input to search in IR based concept location. There are two way to formulate query. One is manually query generation and another is automatically query generation by IR system. To manually generate query the developers has to have previous knowledge. A case study conducted by Marcus [8] showed that automatically generation of query provides better result than manually query. Marcus [13] proposes a recommender (called Refoqus) based on machine learning, which is trained with a sample of queries and relevant results. Then, for a given query, it automatically recommends a reformulation strategy that should improve its performance, based on the properties of the query.

## 3.7 Concept Location in Object Oriented Code

It is well accepted that Concept Location techniques are essential for maintenance of complex procedural code like C. On the contrary, Object

Oriented (OO) Code is structured into classes. A well-designed class implements a single concept. So, it seems that OO code provides advantage for concept location. But Marcus and Rajlich [14] conducted a case study on two Object Oriented softwares named Art of Illusion (AOI) and Doxygen to test the hypothesis “Concept Location Techniques are needed for OO code and OO does not provide advantage for Concept Location”. The case study failed to reject the hypothesis. The study showed that OO does not provide any particular advantage for concept location. A class may represent a concept but a concept is implemented on a single class. May be, the concept is delocalized into several classes for example the concept is delocalized among GUI class and program logic. So, Concept Location technique is necessary for OO code too.

## 3.8 Threads related with Concept Location

There are other threads of research and Software Engineering practice that should be mentioned in the context of concept location.

Program comprehension is an essential part of software evolution and software maintenance [15]. Software that is not comprehended cannot be changed. Program Comprehension is blocks of concepts [11].

Impact Analysis and Fault Localization are other threads. Impact analysis is to check the impact of a change in a system. To find the place of change, the concept location is used.
And a fault could be considered to be an unwanted concept [11].

The literature review shows that IR-based approach is most exercised concept location technique. Marcus [8] first introduced IR-based approach. The refinement of IR-based approach is proposed in several papers in next years. So, Marcus’s IR-based approach is basic for other IR-based approaches. So, it will be good to first understand the IR-based approach described by Marcus’s research paper. To understand the research paper, an implementation of the research paper by developing a ‘concept location tool’ based on methods and techniques described on that paper seems to be inevitable.

## 4 Proposed Methodology

To develop the ‘concept location tool’ to understand IR-based approach to concept location described on the research paper by Marcus [8], the following methodology will be followed.

Step 1: Extracting software requirements from research paper
Step 2: Analysis the requirements
Step 3: Mapping requirements into software components
Step 2: Develop concept location tool
Step 3: Test concept location tool

# 4 Implementation

## 4.1 Information Retrieval based Concept Location

The concept location tool takes two input from the user. One is source code project where the user wants to search and another is a query by which the user wants to search the collection of source code to find the information he needs. A source project is a collection of java class. The system processes the java classes to convert into corpus. In this corpus, there is collection of documents. The granularity of documents is class. The developers search with a query. The system returns a search result of documents based on query. The user investigates the documents.

There are four steps in concept location tool process (Fig 2). These are as follows.

### 4.1.1 Corpus Creation

In this approach, each java class corresponds to a document. Now the whole project is a collection of documents. A java class has comments. A comment can be a part of a method or that class. Comments in java are two kinds. One is javadoc comment and other is implementation comment. A javadoc comment is written for a method or that class with (/**...*/). Implementation comments are inside of methods and fields of class written with (/*..*/) or (//). Comments are parsed from the java class. A javadoc comment different tag like <code>,<h1>, @see, @deprecated are ignored in building the corpus. A comment is treated like natural language processing. The stop words like (i,k,l,and, of, the,etc.) are also ignored. The corpus also ignored the terms starting with constant or a constant. After this, the identifiers from the java class are extracted. The identifiers are from field declaration and method body without comments. The conventional java identifiers for method, variable is camel case(example toString() for a method )and for class is pascal case (example HelloWorld). The developers also use underscore (_) in their methods or variable declaration (example MAX_HEIGHT). The identifiers are added to corpus. The splitting [] terms from identifiers are also added to corpus. Identifiers named like i, j, etc. are also ignored.

### 4.1.2 Indexing

After corpus creation, each class corresponds to a document of the corpus. The corpus has n number of documents. If the corpus is C, then C = $\{d_1, d_2, d_3, d_4, \dots, d_n\}$. Each document di has a collection of terms. Each term in the document has also a term-frequency.

Using Vector Space Model, the documents are indexed into n number of vector documents V = $\{ v_1, v_2, v_3, \dots, d_n\}$. Each document $d_i$ has a corresponding vector $v_i$.

Using Latent Semantic Indexing, The documents are indexed into n number of LSI documents L. The terms are also indexed.

### 4.1.3 Query generation

In both Vector Space Model and Latent Semantic Indexing, the user can manually generate a query from a change request or bug report. In LSI, a query can be generated automatically from the LSI space.
When user generates a query, the stop words from query are removed. For example, a query is “how to increase payment”. Here “to” is a stop word. The words are also stemmed [17], this is for treating the words like connect, connecting, connected as same word as connect.

### 4.1.4 Search Results and Examine

Both in Vector Space Model and Latent Semantic Indexing IR methods, each query is treated as a document $d_q$. The query document is converted into a vector $v_q$. Then the similarity between query and each vector document in the corpus is calculated by Equation 2.4. After calculating the score, the documents of the corpus are sorted descending order by cosine score. The user examines the returned documents.

## 4.2 Information Retrieval with Relevance Feedback based Concept Location

The developers generate a query manually or automatically. The previous mentioned IR based concept location returns a search result based on query. When examining the search result, the developers examine the search results. If the developers don't find the expected document, the developers can give the feedback for better search result. They can mark a document from returned documents as relevant, irrelevant. There are several rounds. In each round, the developers examine the documents and decide whether to stop or go for next round with feedback (Fig 3).
The steps are as follows:

### 4.2.1 Corpus Creation

Same as mentioned at 4.1.1

### 4.2.2 Indexing

Same as mentioned at 4.1.2

### 4.2.3 Query generation

Initial query is generated by the user. After a feedback, the query vector is updated by Rocchio Algorithm (2.7).

### 4.2.4 Search Results

Same as mentioned at 4.1.4

### 4.2.5 Examining Results

The developer examines the search result. If he found the expected document, the concept location succeeds. Otherwise, the developer can go for relevance feedback. He can mark a document from the returned documents as relevant or irrelevant. Based on the feedback, Step is executed.

## 4.3 Languages and IDEs and Version Control System

| | |
|---|---|
| Languages | Java 1.8 |
| IDEs | Eclipse Juno |
| Version Control | Git,<br>Bitbucket |

Table 1: Languages, IDEs and Version Control Systems

## 4.4 External Libraries Used

1. aliasi-lingpipe-4.1.0.jar - SVD Matrix calculation for Latent Semantic Indexing
2. javaparser-core-2.0.0.jar - Parsing Java Source

# 5 Result and Analysis

## 5.1 Studied Project

To evaluate the performance of the concept location tool, Result Analysis Tool (RAT) [19] is selected. RAT is a tool used for examining OMR answer sheet. This tool examines scanner image of OMR sheet, constructs database, data correction, marking, report generation, and result delivery. It is a project developed by the students of Institute of Information Technology. The project is written in java. The project is chosen because it has a complete SRS documentation. The project has a collection of 77 classes (Table 2).

| Project | Description | SRS | #Source Files |
|---|---|---|---|
| RAT | RAT is a tool used for examining OMR answer sheet | SRS of RAT[19] | 77 |

Table 2: Selected Project (RAT)

## 5.2 Data Collection

The features and corresponding implemented classes are extracted from the SRS documentation. From these features, the query represented that feature is generated manually.
The top 11 features are collected as bellow:

| Feature | Feature | Implemented Class(s) | Query | Query |
|---|---|---|---|---|

| No. | | | No. | |
|---|---|---|---|---|
| 1 | Image File Choose | com.resultAnalysisTool.util.fileChoose.FileChooser.java com.resultAnalysisTool.lib.jexplore.FileExplorer.java | 1 | Choosing an image file from explorer |
| 2 | Image Area Defining | com.resultAnalysisTool.tools.imageProcess.areaDefine.ImageAreaDefinerGui.java com.resultAnalysisTool.tools.imageProcess.areaDefine.IndicatorArea.java com.resultAnalysisTool.tools.imageProcess.areaDefine.OptionArea.java | 2 | Define the image area in scanned image |
| 3 | Data Extraction from image file | com.resultAnalysisTool.tools.imageProcess.dataExtract.ImageDataExtractorGui.java | 3 | Extracting data from an image file |
| 4 | Produce result | com.resultAnalysisTool.tools.resultProcess.ProcessResultSelectFiles.java com.resultAnalysisTool.tools.resultProcess.ProcessResultSelectFilesGui.java | 4 | Produce result |
| 5 | Report production | com.resultAnalysisTool.tools.reportCreate.reportGenerate.ReportGenerator.java | 5 | Report production |
| 6 | Report design | com.resultAnalysisTool.tools.reportCreate.reportDesign.ImageArea.java com.resultAnalysisTool.tools.reportCreate.reportDesign.ReportDesignerGui.java | 6 | Report design |
| 7 | Report print | com.resultAnalysisTool.tools.reportPrint.ReportPrinter.java com.resultAnalysisTool.util.filePrint.FilePrinter.java com.resultAnalysisTool.tools.reportPrint.ReportPrinter.java | 7 | Print report |
| 8 | email report | com.resultAnalysisTool.tools.emailPost.EmailSender.java com.resultAnalysisTool.tools.emailPost.EmailSenderGui.java | 8 | Email report |
| 9 | file io operation | com.resultAnalysisTool.util.fileReadWrite.FileIO.java | 9 | File io operation |
| 10 | image | com.resultAnalysisTool.lib.egami.matrix. | 10 | Image matrix |

|  |  |  |  |  |
|---|---|---|---|---|
|  | matrix operation | MatrixUtil.java com.resultAnalysisTool.lib.egami.matrix.Matrix.java |  | operation |
| 11 | font choosing | com.resultAnalysisTool.util.fontChoose.FontChooser.java com.resultAnalysisTool.util.fontChoose.FontChooserGui.java com.resultAnalysisTool.util.fontChoose.TextFont.java | 11 | Choose font |

Table 3 Top eleven features and features implemented into files

## 5.3 Evaluation Matrix

To evaluate the performance of the concept location tool, Top N ranked approach is used.

### 5.3.1 Top N Ranked Approach

If the files associated with features or bugs are ranked in Top N (N =1, 5, 10), the score of Top N is one. If one of the associated is in rank 4, then Top1 = 0, Top5 =1, Top10 = 1. So, from the total number of features or bugs, total Top1, Top5 and Top10 is calculated. The higher value in metric indicates better performances. Top N ranked approach [18] is a popular metric for feature and bug localization.

## 5.4 Result and Analysis

Table 4 and Table 5 show result returned by the concept location tool in Top N ranked approach. A feature is implemented one or more class. When there are multiple classes, the minimum rank of the classes is accepted as final rank of the feature. From Fig 6, it shows that VSM performs better than LSI. In Top 10 rank, VSM performs 73% correct
(Table 5).

| Feature Query No. | VSM Ranking | LSI Ranking |
|---|---|---|
| 1 | 6 | 38 |
| 2 | 9 | 2 |
| 3 | 42 | 49 |
| 4 | 2 | 13 |
| 5 | 16 | 32 |
| 6 | 3 | 8 |
| 7 | 1 | 33 |
| 8 | 1 | 19 |
| 9 | 18 | 38 |
| 10 | 0 | 4 |
| 11 | 2 | 7 |

Table 4: The ranking of desired documents of feature query

|  | VSM | LSI |
|---|---|---|
| Top1 | 0.091 | 0 |
| Top5 | 0.5454545455 | 0.1818181818 |
| Top10 | 0.727 | 0.3636363636 |

Table 5 : The percentage of Top1, Top5, and Top10 ranked documents of VSM and LSI

**Table 4.5 :**

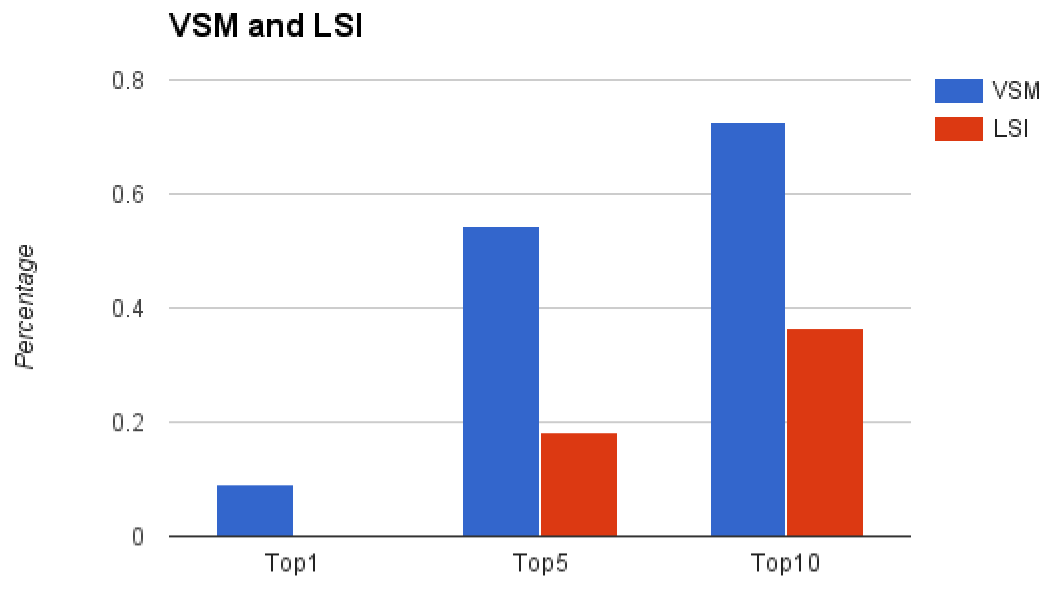

Fig 6 : The comparison between VSM and LSI method

Table 6 contains the results of three rounds of relevance feedback with same set of feature query. Relevance Feedback performs much better than VSM. In Top 10, RF performs about 90% correct where VSM performs 73%. In Rank 1, RF performs much better (55%) than VSM (9%) (Table 6, Fig 7).

| Feature Query | RF round = 0 (rank) | RF round = 1 (rank) | RF round = 2 (rank) |
|---|---|---|---|
| 1 | 6 | 0 | 0 |
| 2 | 9 | 4 | 0 |
| 3 | 42 | 60 | 59 |
| 4 | 2 | 7 | 1 |
| 5 | 16 | 0 | 0 |
| 6 | 3 | 3 | 0 |
| 7 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 |
| 9 | 18 | 4 | 6 |
| 10 | 0 | 0 | 0 |
| 11 | 2 | 0 | 0 |

Table 6: The ranking of three rounds returned by Relevance feedback

| | **VSM** | **VSM_RF** |
|---|---|---|
| **Top1** | **0.091** | **0.5454545455** |
| **Top5** | **0.5454545455** | **0.7272727273** |
| **Top10** | **0.727** | **0.9090909091** |

Table 7: The percentage of VSM and RF

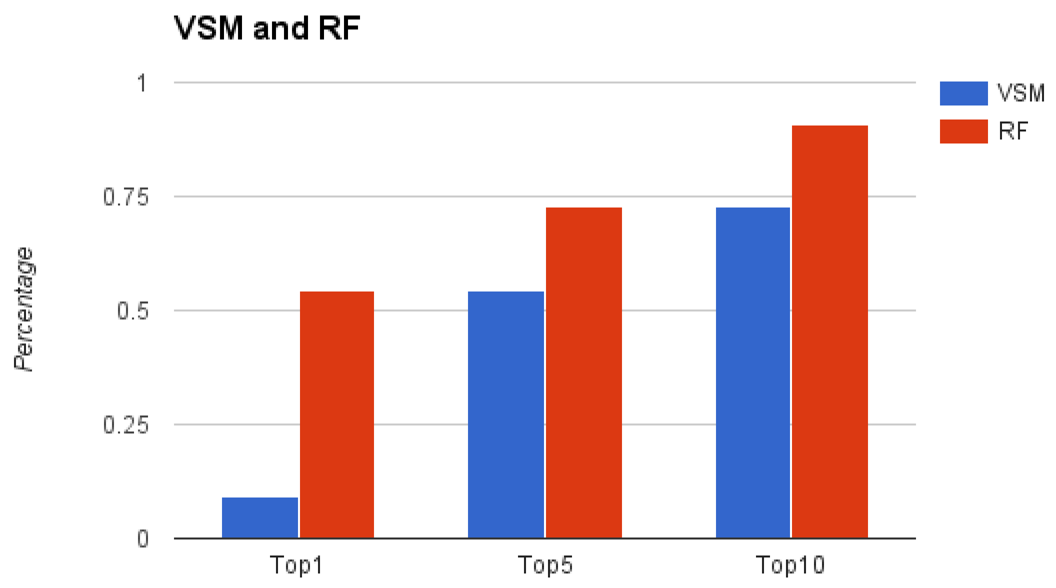

Fig 7: The comparison between VSM and Relevance Feedback

In feature query no. 3, both VSM and RF fail to perform well. Because the initial query “extracting data from image file” is not good. The relevance feedback performs better when the initial query is good.

# 6 Future Work and Conclusion

The concept location tool with IR approach reduces the search space for the developers to look for a specific concept. Relevance Feedback approach performs better than IR methods (VSM and LSI). This concept location tool will be integrated as eclipse plugin.

IR performs better when user query is good. So, the future work will be focused on query generation, processing or understanding.

# 7 References

[1] T. K. P. W. F. a. D. L. Landauer, "An introduction to latent semantic analysis," *MLA,* pp. 259-284, 1998.

[2] K. a. V. R. Chen, "Case Study of Feature Location Using Dependence Graph," *MLA,* 2000.

[3] G. &. M. A. Scanniello, "Clustering support for static concept location in source code," *Program Comprehension (ICPC), 2011 IEEE 19th International Conference,* pp. 1-10, 2011.

[4] G. H. S. M. A. &. M. Gay, "On the use of relevance feedback in IR-based concept location.," *Software Maintenance, 2009. ICSM 2009. IEEE International Conference,* pp. 251-360, 2009.

[5] D. a. A. M. Poshyvanyk, "Combining formal concept analysis with information retrieval for concept location in source code," *15th IEEE International Conference on Program Comprehension (ICPC'07). IEEE,* June 2007.

[6] D. M. G. a. A. M. Poshyvanyk, "Concept location using formal concept analysis and information retrieval," *ACM Transactions on Software Engineering and Methodology (TOSEM) 21.4 ,* 2012.

[7] T. J. B. G. M. a. D. E. W. Biggerstaff, "Program understanding and the concept assignment problem," *Communications of the ACM 37.5,* pp. 72-82, 1994.

[8] A. S. A. R. V. &. M. J. I. Marcus, "An information retrieval approach to concept location in source code," *In Reverse Engineering, 2004. Proceedings. 11th Working Conference on IEEE,* pp. 214-223, 2004 November.

[9] N. a. M. C. S. Wilde, "Software reconnaissance: mapping program features to code," *Journal of Software Maintenance: Research and Practice 7.1,* pp. 49-62, 1995.

[10] V. N. W. M. B. a. H. P. Raijlich, "Software cultures and evolution," *Computer 34,* pp. 24-28, 2001.

[11] V. a. N. W. Rajlich, "The role of concepts in program comprehension.," *10th International Workshop on IEEE, 2002,* pp. 271-278, 2002.

[12] T. R. K. a. D. S. Eisenbarth, "Locating features in source code," *IEEE Transactions on Software Engineering 29.3 (2003),* pp. 210-224, 2003.

[13] S. G. B. A. M. R. O. A. D. L. a. T. M. Haiduc, "Automatic query reformulations for text retrieval in software engineering," *In Proceedings of the 2013 International Conference on Software Engineering,* pp. 842-851, 2013.

[14] A. V. R. J. B. M. P. a. A. S. Marcus, "Static techniques for concept location in object-oriented code," *13th International Workshop on Program Comprehension (IWPC'05), IEEE,* pp. 33-42, 2005.

[15] A. a. A. M. V. Von Mayrhauser, "Program comprehension during software maintenance and evolution," *Computer 28.8 (1995),* pp. 44-55, 1995.

[16] Salton, G. and McGill, M., Introduction to Modern Information Retrival, McGraw-Hill, 1983.

[17] http://tartarus.org/~martin/PorterStemmer/, last accessed Nov 5, 2016

[18] Zhou, Jian, Hongyu Zhang, and David Lo. "Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports." 2012 34th International Conference on Software Engineering (ICSE). IEEE, 2012

[19] https://github.com/MinhasKamal/ResultAnalysisTool, last accessed Nov 8, 2016