# Accelerating Preliminary Analysis of Software Reengineering Process

Kunwar Dev Singh, Saurabh Kumar Singh , Shaganpreet Singh Grewal

Computer Science & Engineering Department,
University of Texas, Arlington
kunwardev.singh@mavs.uta.edu
saurabh.singh4@mavs.uta.edu
shaganpreetsing.grewal@mavs.uta.edu

**Abstract**—*The paper provides an overview of a system to detect the technology stack of a project. When the project is handed to a developer without any implementation, there is no idea of which is the best tool or the environment to set-up the system and further develop the system. Retrieval of the structure and the technology used by the system improves and quickens the reengineering process. In some projects, some various libraries might be missing and project may have certain errors. The tool will allow the developer to detect those files and import them in order to make the project run properly. The algorithm discussed below, makes use of the existing or given project, intakes the files and try to detect the type of project, eg. Web-project, Java project etc., as it matches the various types of projects by observing its files and subfolders from the knowledge base which is linked with the technology stack and returns the technology used in the given project. It tells the user the technology used fot development and also to let the user know about the various files that are required to be imported in order to make the system work properly. This is highly helpful tool, as when used, will reduce the costs involved significantly, especially reducing the time involved in retrieving the structure and the tool, fastening the process of re-engineering. Introducing further upgrades to the tool, it would consolidate its viability and allow the user to use develop the system in a more efficient manner. Industrially the system has great viability, allowing the manager to have better and a more efficient approach towards management of the team as well as the time and cost.*

**Index terms:** Technology Stack, Structure Retrieval, Tool Retrieval

## 1. INTRODUCTION

In the industry, whenever someone is working on a code which has been developed by some other team or developer, one does not have much idea of how the system has been developed by the user. The user/developer needs to spend some time understanding the basic idea of the system and how the system has been structured. Using various tools, if made available by the organisation, helps them to retrieve the basic data flow of the system. But, the user still needs to apply his coding skills to get hold of the entire system's functioning. Even after retrieving the various artifacts related to the system, user still has a huge task of importing the project into some tool and upgrading the system or making any changes to the system.There are various tools which can come handy in doing the project. But there are certainly some tools that are better suited or has better environment setting, in which the project can be re-engineered in a better way or further developments can be made in a much easier way.

Retrieval of missing files is one of the most critical steps in order to implement the changes. The project if not imported properly, that is, with all the dependent files and variables, there will be many issues. The missing/dependent files retrieval helps the user in that case. The procedure of how it retrieves the files has been explained in details in the further sections of the paper. The detection procedure let the user know which files that are necessary for the project are missing from it.

The paper explains all the various elements that have been considered while elaborating the tool. The contents of the research paper explains the aforementioned tool with the help of algorithm designed, various diagrams such as use case diagrams, data flow diagram and others. The paper also explains the theory of how the tool is interacting with the project and how it is returning the technology stack from the knowledge base, technology in which the project has been developed so far.

The first part involves how the tool makes use of the missing file detection system, to retrieve all the missing files. Once, the project is loaded into the system I, it uses the file detection algorithm to complete the project. Next, paper talks about the basic algorithm on which the suggested tool is based on. The algorithm gives an outline of the system, of how the system takes a Java project as input, applies the algorithm so as to retrieve the structure of project and applies the missing file detection and further retrieve the required instances such as structure, best environment and the tool best suited to develop system further and incorporate changes to it. The missing files are retrieved in the first part which are incorporated further into the main working of the function.

## 2. DESCRIPTION

There is a high demand of a tool or a software, which can help the developers in identifying the structure and main the main functionalities of any project. If any developer is given a code by any other user than there is a huge amount of resources that are required in order to build and run the given project. In that sense one require to have the tool that can provide the developer with all the functionalities, that are necessary in the reengineering process and also in the further development and upgrades of the system.

The major field where the tool targets to provide support to developers is of Java. Java is one of the most popular environment to develop any project. The java developer accounts for the highest number of developers in any field of specialisation. But, the life of any developer is not easy at all and same goes for java developers as well. While developing projects, and also in case of upgrading a pre-designed system, The main idea behind targeting the java projects, is that it is one of the most widely used technology in all the organisations all over.

The very first problem that may occur, is that the system or project may be missing any required dependent that are required in order for the system to work properly, or for that case, even build and run. In order to detect and import all those missing files, one may have to use various softwares, just to do this job. Other option that developer may opt for is that of trying to import all the files manually, downloading via the internet. This is not a very useful idea as it wastes a lot of resources, especially time.

Another problem that a developer may face is that of improper structure. Project may not be structured properly for the user to implement the algorithms and the required changes. The developer needs to address this issue by studying the whole structure of the code and retrieve the important artifacts. Only after retrieving the structure and artifacts related with how the data flows through the system, only then the developer is able to efficiently implement is ideas to the project.

Thirdly, but may be the biggest problem a developer might face is that of selecting the best suited environment and the tool for that particular project. The developer may know the technology but may not know the best suited environment and tool for it. To suitably find the tool, one may have to go through all the various tools possible. This may take most of the resources that are available for the user at that time to implement the whole system incorporating further changes. It is not very a good methodology to use while re-engineering any system as it may hamper the whole process of development.

The tool is defined for these specific cases, to handle all the various problems that a developer might face while implementing changes to a pre-designed system and also while implementing reengineering. The tool, which is described with each of its functionalities in further parts, targets to solve all the problems and retrieve all the information that is required by the user.

Rather than using a bunch of various tools for each part of the development process, the tool solves all the problems at once. The tool is based on the idea, that once the project is imported into the tool, the tool will retrieve all the necessary information for the developer to implement. How the data flows through the tool and how the various artifacts are retrieved, is defined in the further sections with the help of various diagrams, such as flow diagram, use case diagram etc.

## 3. MODULES OF THE ALGORITHM

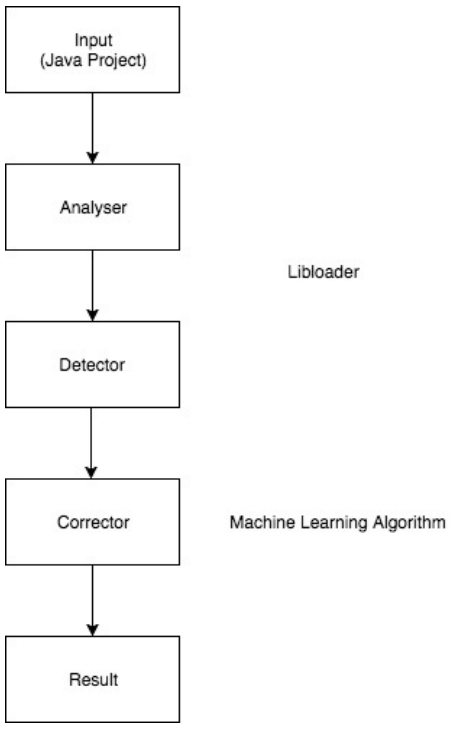


**FIG. 3.1**

The above diagram shows the peripheries of the solution given.

**Input** : The input is the Java project(can be any type of Java project based on IntelliJ, Eclipse and Netbeans) with complete files and folders along with other technologies.

**Analyser** : The analyser checks the structure of the input application and converts it into data structures that can be used by the algorithm that follows. An analysis of the tools and technologies involved is done by reading the project management, properties and other project manifest files.

**Detector** : The detector is a pessimist module that tries to verify the project and find if any correction or modification is required at the current state of the project. Detector is an important part as these findings are used by later modules and also part of the result set.

The Analyser and Detector module is well explained in the section 6 of the paper along with analogy to an existing tool which can either be taken as an example of these modules or as a replacement.

**Corrector** : After the structure issues or missing libraries or any other error is detected, the correction is also handled by the algorithm in this module. Here we look at how to resolve dependencies if possible or at least point these issues in the result.

**Result** : After all the modules of the algorithm have done their part we come to the final step where we display the details of all the module findings and related work done. The errors found, errors resolved and issues that need attention are typical result attributes along with other attributes.

The second module diagram shows the machine learning part of the solution.

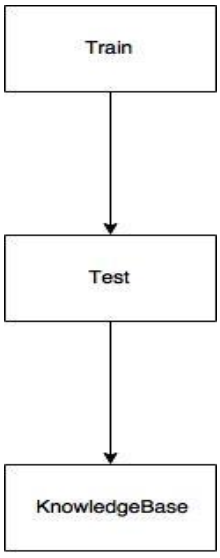


**FIG 3.2**

**Train**: This module is the training part of the Machine Learning system of the paper. It talks about the machine learning algorithm used and worries about the accuracy, relevant training data used as these factors matter the most while considering how well our suggested tool will perform.

**Test**: The test module addresses the question of how well the machine learning algorithm is performing. A good portion of the data is reserved as test data and is used as testing data to know the accuracy of the algorithm. Although this accuracy is by no means the accuracy on the future data(the projects that will be evaluated using our tool) as that accuracy will not be known until the tool goes in the beta phase.

**Knowledge Base**: The knowledge base can be considered as the most critical part of our paper. This is what we are trying to create using as much training data as possible. This is the ultimate repository that will be referred for all future projects before classifying them and identifying their technology stack. The knowledge base will contain all the features captured by our algorithm and using them for classification.

## 4. CONVERSION TO TREE STRUCTURE

This section of the paper talks about how the input project is processed into a tree structure and then, making use of this tree structure we compare it to the technology stack we have got and find the best possible tool and the developing language which should be used to apply reengineering on the project and add further modification to the already developed project.

The initial process of the process includes the project, which needs to examined for the tool, is acts as the input for the algorithm. The project imported is tested upon various properties like the structure of the project. The main purpose of the algorithm is to apply the structure analysis of the project. It gives us a relative idea of how the system has been designed. The structure of the project can be retrieved from the various tools that are already available to be used, such as RFEM, Prokon and many others.

While retrieving the structure of the file, when the tool is parsing through the project, it will create the tree structure according to the subfolders or subdirectories that are linked directly to the root package/folder of the project. This is an iterative process and involves the use of Machine Learning concepts. Machine learning helps to define the type of project user is working on, by comparing it to Knowledge Base, which is described in detail in section 7.

When the tool is parsing through the project folder, it creates the subdirectories for all the subfolders or subdirectories it encounters. So the main project is the root folder for the tree. When the next subfolders are detected by the tool, it represents them as the child of the root folder. It parses all the subfolders of the root directory and detects all of them and add them as child nodes of the root of the tree. The tool then tries to match if any of the child nodes belong to any of the basic projects in the knowledge base. If it detects any of the files that are present in the knowledge base, that is the files with the same extension, those files are added into the root folder of the project and marks it as one of the types which the project might be. This is an iterative process and it runs till all the files of the project has been parsed, and no node in the tree has a further child node that is present in the project structure.

While parsing through the whole project, whenever a file having the same extension of the file present in the

knowledge base, that is the data retrieved from the training the algorithm on huge number of projects, those are the types the project is likely to be. All the file extensions that are retrieved from the parsing through the project, those are the likely types that the project might, as there can multiple types of extensions retrieved, so the project might be a combination of those all. The following diagram represents the process of how the files and types are retrieved from the main project.

FFug

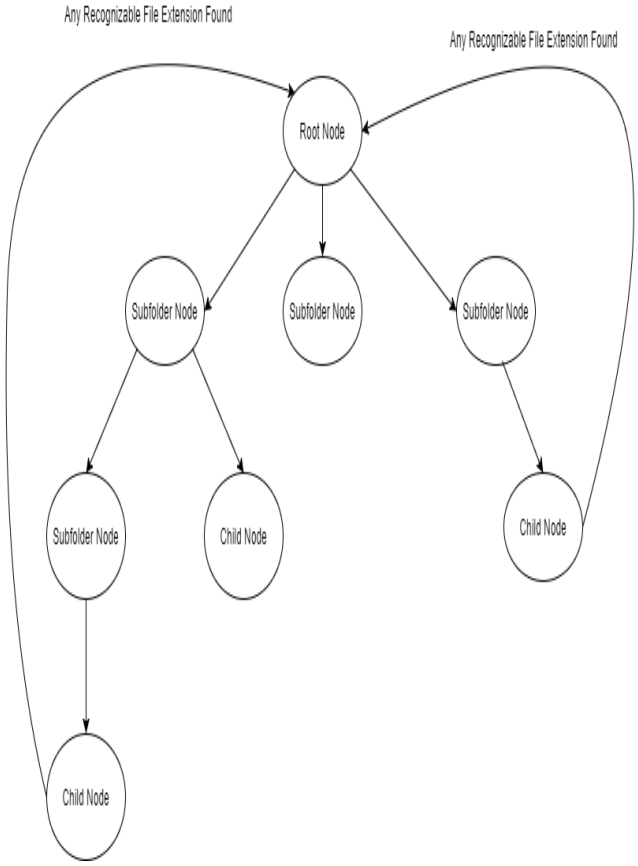


**FIG 5.1**

To further explain this whole process of conversion to tree structure and the process of file retrieval, let's say we have the following data to apply our tool on.

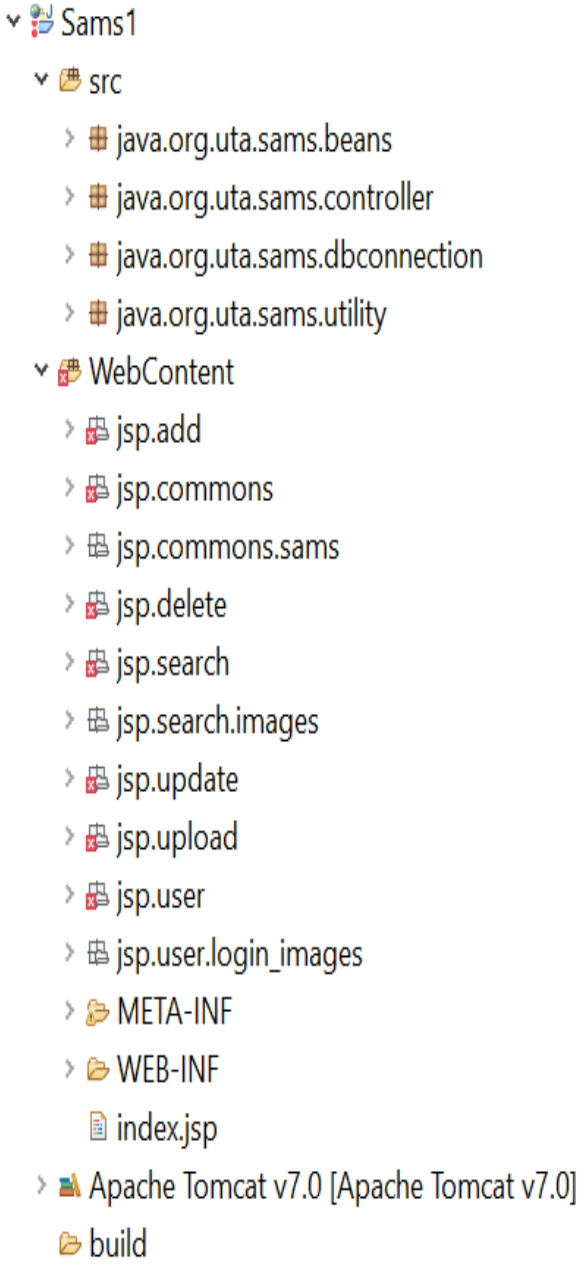


**FIG 4.2**

When this project is imported into the tool, the tool create a root node for the project in SAMS, which acts its root node. the tool then parses the next subfolder of the project folder. It is found that there are four subfolders for the root node. It then creates 4 child nodes to that root node. The tool then parses the further root nodes to check for any further subfolder or sub-files. The process is continued till all the folders have been parsed and there is no subfile or subfolder present in the project that has not been parsed yet. After applying the above algorithm/process on the project shown in the figure, the tree structure will look similar to the

following tree shown in the figure.

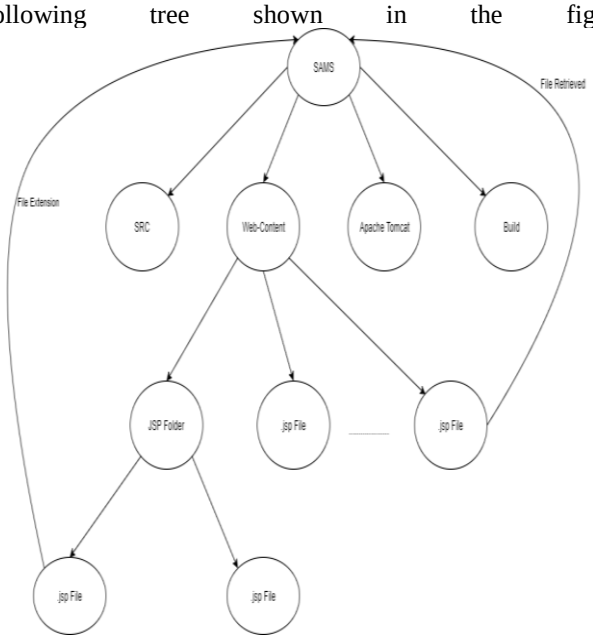


**FIG 4.3**

Another important retrieval from this is of feature capturing. While implementing this step, we are parsing each file to create the tree and hence each node of the whole project is covered. In doing so, each file is processed to capture all the features that each file is implementing. All the features are then accumulated by the end the processing of the project is completed.

## 5. THE ALGORITHM

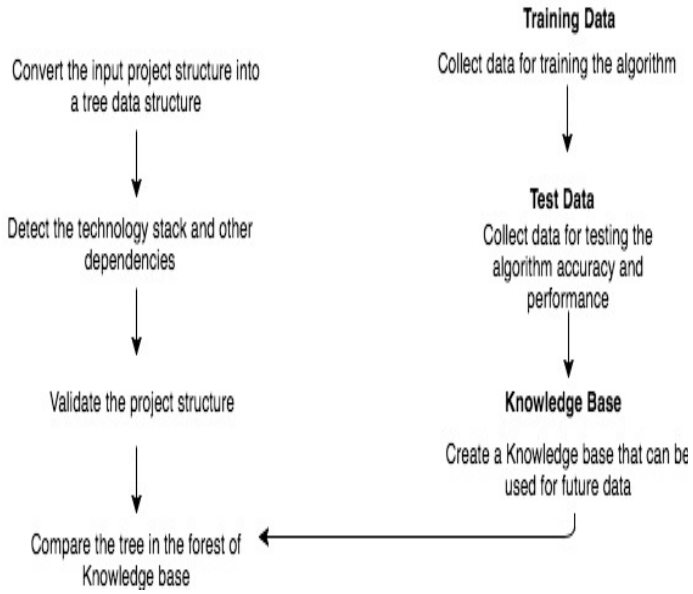


**FIG. 5.1**

The algorithm section of the paper is divided into two parts: The Machine Learning Algorithm and the Java Environment Algorithm.

**The Machine Learning Algorithm** : This algorithm is how the knowledge base is created for future Java project matching as part of detection. The knowledge base is define
Training Data : The data that contains large number of Java projects using which classification is done into different classes. During this step the algorithm discovers the data and does clustering if required.
Test Data : The test data is used to get the efficiency of the Algorithm and also the performance factor. This can be used as a feedback for the algorithm that can be used for further improvements and evaluation between different Machine Learning Algorithm.

**The Java Environment Algorithm** : This algorithm is the main algorithm that handles matching of the input project with the projects that are part of our learning(knowledge base). This algorithm has been made such that it will try to get best out of the KB(knowledge base) and will detect what tools have been used or how much it matches with classes that have been trained in the Machine Learning algorithm and to which extent does a match exist.

## 6. USE CASE


**FIG 6.1**

Fig. 6.1 represents the use-case based on the tool. It shows how the interaction between the user and the tool takes place. There is only one user who will use the tool by importing the project into the tool. The next part of the procedure includes the way tool processes the project imported.

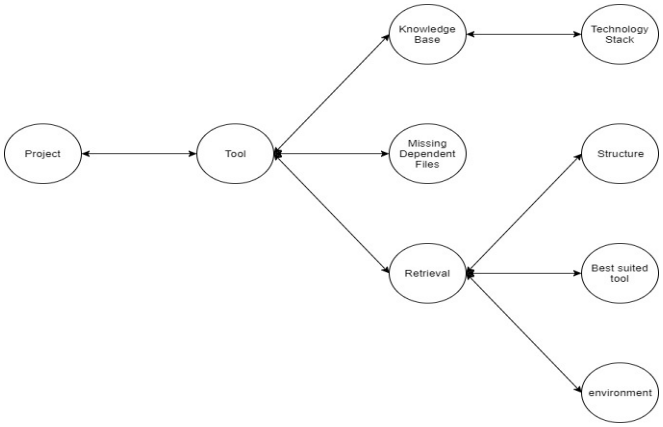


**FIG 6.2**

Fig. 6.2. represents the flow diagram for the tool. it explains how the flow of the data takes takes places between the different sections of the tool. The above diagram in short explains how the various functions are interrelated with each other. It describes what are the various functionalities that the system is currently executing .

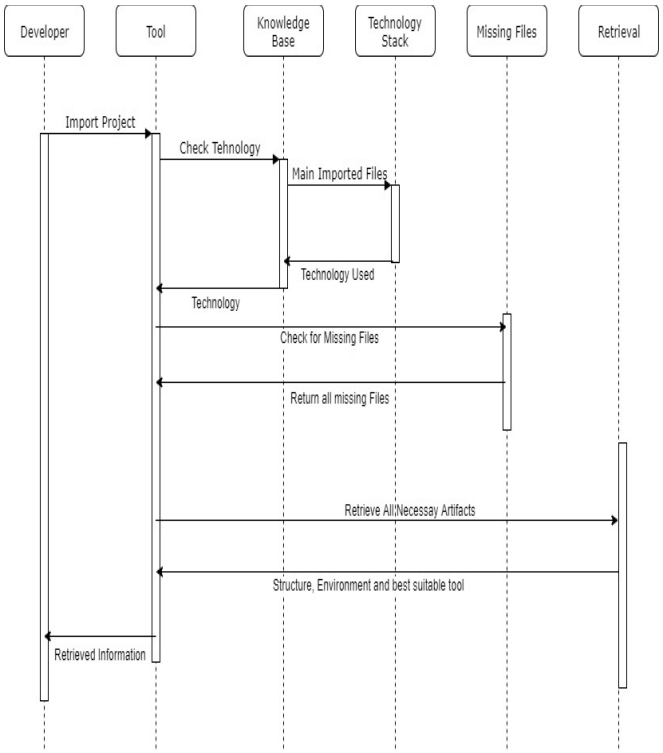


**FIG 6.3**

Fig. 6.3. represents the sequential diagram for the user interacts with the system. This diagram shows exactly how the procedure takes place. Firstly, after importing the system into the tool, the tool retrieves from the technology stack in which technology the system has been developed. After retrieving the technology, the tool then tries to detect if any of the dependent files are missing from the project. If yes, then it runs the algorithm to retrieve all those files. Now the technology has been identified as well as the missing files, now the main part is to retrieve the major artifacts in order to aid the developer in further upgrading the system. In the next step, it retrieves the structure of the project, which is the best tool to make further changes and finally, which is the best environment to set up the project. In these three steps, the tool retrieves the major information and finally returns as an output to the user.

## 7. Handling the Missing Libraries in JAVA

There are many commonly used libraries across a selection of JAVA program. There are more specifically evaluating execution details for each library. Although analysis included over 2000 projects, resolution of missing, referenced libraries was not automated.

### 7.1 Recovering Origin of Software

Using a set of techniques like signature matching, source code fact extraction, software clone detection for identification. However, it needs a lot of resources, making it impractical for the process to execute. Muse Understand Scripting Engine (MUSE) is another typical example of a source code based tool developed by us that measures the object oriented design best practices.

### 7.2 LibLoader

We suggest that the tool used for the part of the algorithm that handles the missing libraries present in the JAVA project. This tool is called LibLoader. MUSE (mentioned above) does not extract metrics directly from source code but uses commercial tool UNDERSTAND for source code analysis. Based on the information provided by it, the identification of missing classes, packages, and types is possible. This enables determination of missing libraries and completion of source code.

The process that LibLoader executes is explained in the following part:

#### 7.2.1 DATABASE CREATION

It extracts meta-information from source code file and stores this information in a database. The meta-information comprises type-definition, class relations and method

definitions. This is the input for MUSE analysis. It control the process of database creation by monitoring output and logging information of Understand which is used to detect missing library.

### 7.2.2 IDENTIFICATION

This step will identify the missing types and classes of the project. It issues the warning and error message by issuing name, type and class. This information is parsed and the list is made. For the retrieval of the missing library, this tool currently rely on Maven Central Repository using provided REST-API. The search result contains Group ID, Artifact ID and version of missing libraries that are found. It stores the first 5 libraries of each request. It uses Maven Build manager to download necessary libraries.

It adds downloaded libraries to build path of Understand Project and Understand re-analyzes the project. It adds downloaded library to build path

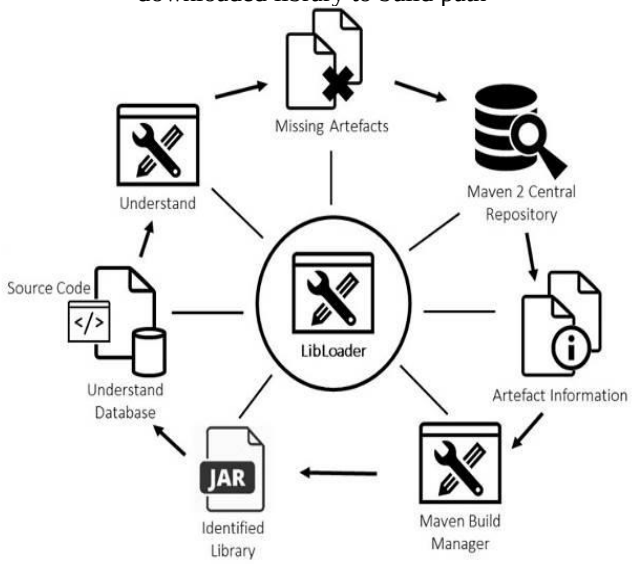


**FIG 7.1**

Fig. 7.1 depicts repetitive nature of analysis process. Adding 1 library per iteration is more time consuming than adding multiple libraries per iteration. By default, LibLoader downloads release version of each relevant library. It is able to replace already added libraries by another version of same library.

By default, LibLoader downloads the release version for each relevant library. Although the library is added to the project, there still may exist missing artefacts, identifying an already added library as missing. The existence of types, classes, or packages within libraries may differ from version to version. Because of such version incompatibilities, missing types occur. LibLoader is able to replace already added libraries by another version of the same library. This feature increases the probability of correct versions of the added libraries and reduces the number of unresolved missing artefacts.

## 8. THE KNOWLEDGE BASE

The knowledge base is the master repository using which the future projects will be classified. This classification will be done using Machine Learning supervised learning. The knowledge base is a critical part for the machine learning module. In the file tree structure construction we capture the type of files in the project and other details which can be considered as the feature capturing step. This step when followed for thousands of training instances gives birth to a repository that we refer to as the knowledge base. This can be considered as a directory that stores all the features and also the class that they belong to. The more the data stored the more accurate the classification will be. Our main aim in context to improving the efficiency of the system should be to improve this module.

## 9. Naive Bayes Classifier

This is the algorithm that we are going to use. It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'. Before explaining the algorithm, there is a need to clear the basics, which we are covering in the upcoming sections.

### 9.1 Probability Theory

This is the branch of mathematics concerned with probability. Although, there are several different probability interpretations, probability theory treats concept in rigorous mathematical manner by expressing it through set of axioms. It gives us a framework to model these decisions which help in making them more efficiently. They help us make decisions where there are several different probability interpretations, probability theory treats concept in rigorous mathematical manner by expressing it through a set of axioms. They help us make decisions when we have perfect informations. It trains us to make decisions where there are indeed observable patterns but also degree of uncertainty. There are different types of probabilities which are used:

1. **Conditional Probability**: It is the measure of the probability of event, given that (by assumption, presumption, assertion or evidence) another event has occurred.[1] If the event of interest is *A* and the event *B* is known or assumed to have occurred, "the

conditional probability of $A$ given $B$", or "the probability of $A$ under the condition $B$", is usually written as $P(A|B)$, or sometimes $P_B(A)$.

2. **Joint Probability:** A joint probability is a statistical measure where the likelihood of two events occurring together and at the same point in time are calculated. Joint probability is the probability of event Y occurring at the same time event X occurs.Notation for joint probability takes the form: P(X∩Y) or P(X and Y) or P(XY), which reads as the joint probability of X and Y.
3. **Marginal Probability:** In probability theory and statistics, the marginal distribution of a subset of a collection of random variables is the probability distribution of the variables contained in the subset. Marginal variables are those variables in the subset of variables being retained. These concepts are "marginal" because they can be found by summing values in a table along rows or columns, and writing the sum in the margins of the table.

### 9.2 Bayes Theorem

In probability theory and statistics, **Bayes' theorem** (alternatively **Bayes' law** or **Bayes' rule**) describes the probability of an event, based on prior knowledge of conditions that might be related to the event. One of the many applications of Bayes' theorem is Bayesian inference, a particular approach to statistical inference. When applied, the probabilities involved in Bayes' theorem may have different probability interpretations. With the Bayesian probability interpretation the theorem expresses how a subjective degree of belief should rationally change to account for availability of related evidence. Bayesian inference is fundamental to Bayesian statistics.

### 9.3 Different Interpretations

The interpretation of Bayes' theorem depends on the interpretation of probability ascribed to the terms. The two main interpretations are described below:

#### 9.3.1 Bayesian Interpretations

In the Bayesian (or epistemological) interpretation, probability measures a "degree of belief." Bayes' theorem then links the degree of belief in a proposition before and after accounting for evidence. For example, suppose it is believed with 50% certainty that a coin is twice as likely to land heads than tails. If the coin is flipped a number of times and the outcomes observed, that degree of belief may rise, fall or remain the same depending on the results. For proposition $A$ and evidence $B$,

- $P(A)$, the *prior*, is the initial degree of belief in $A$.
- $P(A \mid B)$, the *posterior* is the degree of belief having accounted for $B$.
- The quotient $P(B|A)/P(B)$ represents the support $B$ provides for $A$.

#### 9.3.2 Frequentist Interpretations

In the frequentist interpretation, probability measures a "proportion of outcomes." For example, suppose an experiment is performed many times. $P(A)$ is the proportion of outcomes with property $A$, and $P(B)$ that with property $B$. $P(B \mid A)$ is the proportion of outcomes with property $B$ *out of* outcomes with property $A$, and $P(A \mid B)$ the proportion of those with $A$ *out of* those with $B$. The role of Bayes' theorem is best visualized with tree diagrams, as shown to the right. The two diagrams (Fig 3) partition the same outcomes by $A$ and $B$ in opposite orders, to obtain the inverse probabilities. Bayes' theorem serves as the link between these different partitionings.

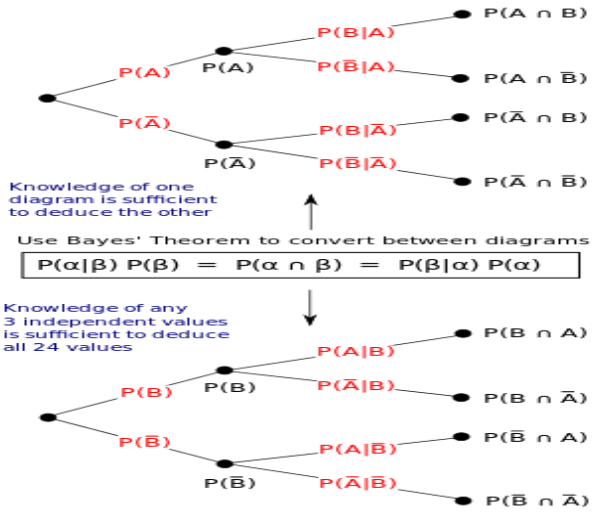


**FIG 9.1**

### 10. Method

Our plain is to extract at least 5000 eclipse project that will be used for training our model. A total of eclipse projects which are first taken into consideration for the initial stage:

1. JAVA Project
2. JAVA EE Application Project
3. Enterprise Application Project
4. JAXB Project
5. Dynamic Web Project
6. Static Web Project
7. JavaScript Project
8. Maven Module

9. Maven Project

Our main aim will be to segregate all these eclipse projects and put them in different folders and segregate them into different classes. Our program is divided into further parts which are explained below.

### 10.1 Training the Data

We will apply the tree constructing phase on each project. That function will save the file in metadata. Since, it will extract the file format that are present in the folders, it will be saved in the csv file that has the heading of each file format. These are the attributes of the file extensions (that is .java, .xml, .jsp, etc) on the basis of which our Naive Bayes Classifier will be trained. The files in a certain folder will be either present (1) or absent (0). This will create our training data.

### 10.2 Testing Phase

During this phase, the given folder will be put in the tree function and converted into tree and put in the test file which is present in a different folder. On that test file, the predict function of the model is applied. This will give us the type of project, which is given. Accordingly, then the lib-loader function is applied on the project, that will be run on the project in order to find the missing libraries and dependencies.

## 11. Future Work

The future work for the paper is vast as this idea can be applied to a numerous places apart from just Java based projects. Below are the sections where is a lot of scope of improvements and scalability.

- The Machine Learning Algorithm can be improved a lot as any other classifier can be. We are using Naive Bayes classifier but other classifiers can also be tried to improve the accuracy not just on the test data but on the future data.
- The mentioned idea can be applied to any other technology in a similar way as has been proposed for Java Eclipse project.

## REFERENCES


**[1] T. Atzenhofer, R. Plösch: Automatically Adding Missing Libraries to Java Projects to Foster Better Results from Static Analysis, 17th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2017), Shanghai, China, September 17-18, 2017.**

**[2] . Nagappan and T. Ball, "Static Analysis Tools As Early Indicators of Pre-release Defect Density," in Proc. of the 27th Intl. Conf. on Software Eng. St. Louis, MO, USA, 2005, pp. 580–586.**

**[3] Gruber, C. Körner, R. Plösch, G. Pomberger, and S.Schiffer, "Benchmarking-Oriented Analysis of Source Code Quality: Experiences with the QBench Approach," in Proc. of the IASTED Int. Conf. on Software Eng. (SE 2008), Anaheim, CA, USA, 2008,pp. 7–13.**

**[4] Bräuer, R. Plösch, and M. Saft, "Measuring Maintainability of OO-Software - Validating the IT-CISQ Quality Model," in Proc. of the 2015 Federated Conf. on Software Develop. and Object Technologies, Žilina, Slovakia, 2015, pp. 283–301.**

**[5] Bauer, L. Heinemann, and F. Deissenboeck, "A Structured Approach to Assess Third-Party Library Usage," in Proc. of the 2012 IEEE Int. Con. on Software Maintenance (ICSM), Washington, DC, USA, 2012, pp. 483–492.**

**[6] Davies, D. M. German, M. W. Godfrey, and A. Hindle, "Software Bertillonage: Determining the Provenance of Software Development Artifacts," J. Emprical Softw. Eng. , vol. 18, no. 6, pp. 1195-1237,Dec. 2013.**

**[7] . Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi,"Sourcerer: Mining and Searching Internet-Scale Software Repositories," Data Min. Knowl. Discov., vol. 18, no. 2, pp. 300–336, Apr. 2009.**

**[8] Ossher, S. Bajracharya, and C. Lopes, "Automated Dependency Resolution for Open Source Software," in Proc. 2010 7th IEEE Work. Conf. on Mining Software Repositories (MSR),Cape Town, South Africa, 2010, pp. 130–140.**

**[9] J. Zhao and N.Badler Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures. ACM Transactions on Graphics, pages 313–336,October 1994.**

**[10] S.R.Clay and J. Wilhelms. Put: Language-Based Interactive Manipulation of Objects. IEEE Computer Graphics and Applications, pages 31–39,March 1996**

**[11] B.Hawkins. The Semantics of English Spatial Prepositions. PhD thesis, University of California, San Diego, SanDiego, CA,1984.**